Data-Driven Methods for Compression and Editing of Spatially Varying Appearance

Dissertation

zur
Erlangung des Doktorgrades (Dr. rer. nat.)
der
Mathematisch-Naturwissenschaftlichen Fakultät
der Rheinischen Friedrich-Wilhelms-Universität Bonn

vorgelegt von

Dipl.-Inf. Gero Müller

aus Köln

Bonn, Dezember 2008

Universität Bonn
Institut für Informatik II
Römerstraße 164, D-53117 Bonn

Angefertigt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Rheinischen Friedrich-Wilhelms Universität Bonn

Dekan: Prof. Dr. U.-G. Meißner

1. Referent: Prof. Dr. Reinhard Klein
2. Referent: Prof. Dr. Volker Blanz
3. Referent: Prof. Dr. Jan Kautz

Tag der Promotion: 1. September 2009
Erscheinungsjahr 2009
Diese Dissertation ist auf dem Hochschulschriftenserver der ULB Bonn
`http://hss.ulb.uni-bonn.de/diss_online`
elektronisch publiziert.

CONTENTS

ZUSAMMENFASSUNG

Das optische Reflexionsverhalten von Materialien und Oberflächen ist für die korrekte Wahrnehmung und Einschätzung einer Szene von großer Bedeutung. Der spezifische Glanz und die Textur einer Oberfläche geben beispielsweise Auskunft darüber, ob ein Material glatt oder porös, weich oder hart ist, oder sogar, ob es im Falle eines Textils aus einer Baumwoll- oder Synthetikfaser besteht. In der Computergrafik wurden diese Aspekte einer Szene bis vor wenigen Jahren noch recht stiefmütterlich behandelt. Dies kann man einerseits darauf zurückführen, dass man sich der Bedeutung von Materialien für die Wahrnehmung von Realismus möglicherweise noch nicht ausreichend bewusst war. Andererseits ist das Modellieren von Materialien aufgrund der beteiligten mikro- und mesoskopisch kleinen und komplexen Strukturen sowie deren nicht-trivialen Einfluss auf das Reflexionsverhalten ausgesprochen schwierig. Dies gilt insbesondere, wenn nur klassische Modellierungstechniken wie parametrische Reflexions- und Oberflächenmodelle zur Verfügung stehen, deren Parameter häufig unintuitiv zu bedienen sind und die unter Umständen gar nicht in der Lage sind, das gewünschte Verhalten zu erzeugen.

An diesem Punkt setzen nun die so genannten *daten-getriebenen* Verfahren an: Anstelle eines parametrischen Modells werden gemessene Daten des zu visualisierenden Objekts verwendet. Beispiele hierfür sind per Laserscanner akquirierte Geometriedaten oder mittels eines Motion-Capture Systems gewonnene Bewegungsdaten. In der Form von fotografierten Texturen sind daten-getriebene Verfahren schon seit vielen Jahren für die effektive Repräsentation komplexer Materialoberflächen im Einsatz.

In dieser Arbeit soll es nun um eine erweiterte Definition von Texturen gehen: die *Bi-direktionale Texturfunktion*, kurz BTF. Im Unterschied zu gewöhnlichen Texturen, die meist nur durch ein einziges Bild repräsentiert werden, besteht die BTF aus vielen Bildern, die unter verschiedenen Beleuchtungen und unter verschiedenen Blickwinkeln aufgenommmen wurden. Somit repräsentieren sie das Aussehen eines Materials unter der Variation von Beleuchtungs- und Betrachterrichtung. Leider findet die BTF in der Praxis bisher kaum Verwendung, da die große Menge von Bildern, welche in der Regel erforderlich ist, um ein Material mit ausreichender Genauigkeit erfassen zu können, schwer zu verarbeiten ist.

In der vorliegenden Arbeit werden nun Verfahren vorgestellt, welche die praktische Anwendbarkeit der BTF deutlich verbessern sollen, indem sie erstens, die zu speichernde Datenmege deutlich reduzieren, und zweitens, das effiziente Editieren der BTF ermöglichen. Dementsprechend besteht diese Arbeit im wesentlichen aus zwei Teilen:

Der erste Teil stellt Techniken vor, welche in der Lage sind, die Datenmenge unter Beibehaltung der visuellen Qualität effektiv auf weniger als ein Prozent der Originalgröße zu reduzieren. Hierbei wird im besonderen darauf Wert gelegt, dass sich einzelne Datenwerte möglichst effizient wieder aus der komprimierten Darstellung rekonstruieren lassen, was für die interaktive Darstellung in Renderingsystemen unerlässlich ist. Die Grundidee der vorgestellten Algorithmen besteht darin, mathematische Verfahren aus dem Bereich der Faktorisierung von Matrizen auf die gemessenen Daten der BTF zu übertragen. Ein weiterer wesentlicher Beitrag dieses Teils der Dissertation ist ein effizienter Algorithmus zur Berechnung von lokalen Koordinatensystemen rein auf Basis der gemessenen Daten. Der Algorithmus kommt im Gegensatz zu früheren Arbeiten zu diesem Thema ohne die Annahme eines spezifischen Reflektionsmodells aus.

Im zweiten Teil geht es dann um Algorithmen, die die Veränderung der Struktur und des Reflexionsverhaltens einer BTF ermöglichen, ohne jedes der vielen tausend Bilder einzeln für sich editieren zu müssen. Unsere Technik zeichnet sich durch die Fähigkeit aus, einfache Editieroperationen, die auf ein einzelnes Bild angewendet wurden, konsistent auf den gesamten Datensatz übertragen zu können. Zu diesem Zweck greift unsere Methode auf Techniken aus dem Bereich der Textursynthese und des Texturtransfers zurück und verallgemeinert diese auf BTFs. Ein weiterer Aspekt unseres Verfahrens ist die sinnvolle Interpolation von BTFs basierend auf verallgemeinertem Texturtransfer. Dies ermöglicht das Editieren der Reflexionseigenschaften von BTFs auf eine vollständig daten-getriebene Art und Weise.

ABSTRACT

The optical reflectance behavior of materials and surfaces is of great importance for the perception of a scene. The specific gloss and the texture of a surface give valuable hints on properties of the material it is made of; like if it is smooth or porous, soft or firm or even, in the case of a textile, if it is made from cotton- or synthetic fibers. During the early years of Computer Graphics these aspects have been slightly overlooked and materials were modeled based on quite simple assumptions about reflectance and texture appearance. Maybe this can be appointed to the fact that at that time the community had not realized the importance of surface appearance in its full extent. Another reason might be that materials are just particularly difficult to model: the involved micro- and mesoscopic structures have a non-trivial influence on the reflectance behavior and appearance. The standard models that have been used for the representation of surface appearance during the first two decades of computer graphics were clearly not sufficient to model this complexity in an effective and intuitive way.

At this point the so-called *data-driven* methods step in: instead of using a parametric model the idea is to base the scene representation on measurements of the object one wants to visualize. Common examples for this data-driven way of image generation are laser-scanned geometry data or animation data, acquired by motion-capture systems. In terms of spatially varying material appearance data-driven methods are already common practice in the form of textures which are usually special photographs of a material.

In this thesis we will deal with an extended definition of classical image textures: the Bi-directional Texture Function (BTF). In contrast to ordinary textures, which usually consist only of a single image, a sampled BTF comprises thousands of images which have been captured under various light situations and from different view points. This way the BTF represents the appearance of a material under varying view- and light direction which explains its defining term *bi-directional*. Due to the large number of images required to capture this behavior with sufficient accuracy the BTF is rarely used in today's practical applications. This dissertation aims at improving the practical applicability of BTFs by introducing novel methods for effective BTF data reduction and editing. Consequently, the thesis consists of two parts:

The first part introduces and compares techniques for reducing the effective size of the data that needs to be stored on disk down to less than one percent of the original size while preserving the visual quality. Special emphasis is laid on efficient reconstruction from the compressed representation which is indispensable for interactive display in rendering applications. The main idea of the presented techniques is to transfer algorithms known from the field of matrix factorization to BTF data. Another substantial contribution of this work is an efficient algorithm for the computation of local coordinate systems from the raw measured reflectance data alone. In contrast to previous methods it does not rely on assumptions about a specific reflection model.

In the second part we will present algorithms which enable editing of the spatial structure and reflectance of the BTF without the need to touch the thousands of images individually. Our technique is characterized by the consistent and interactive transfer of simple editing operations performed on a single image to the whole data. For this purpose, we generalize techniques known from texture synthesis and transfer to BTFs. Furthermore, we introduce meaningful interpolation of different BTFs based on our generalized appearance transfer technique. It allows to intuitively edit their reflectance in a completely data-driven way.

ACKNOWLEDGEMENTS

The famous metaphor of the researcher standing on the shoulders of giants often came to my mind while writing this thesis.

The first giant I would like to thank is my supervisor Prof. Dr. Reinhard Klein who allowed me to set my feet on his shoulders any time. I will deeply remember many fruitful discussions both of us standing in front of the famous white blackboard resulting in many of the findings presented in this thesis. Without him recognizing the potential of image-based material representations this work would have never been possible.

As representatives for the computer graphics research community I am also very grateful to Prof. Dr. Volker Blanz and Prof. Dr. Jan Kautz who kindly agreed to serve as external reviewers.

During the last years I have realized how much luck it is to work in an environment with so many wonderful colleagues as in the Bonn Computer Graphics group and I have to express my biggest thanks to all of them. In particular, I would like to mention Gerhard Bendels, Dirk Koch, Jan Meseth, Martin Rump, Ralf Sarlette and Mirko Sattler with whom I had the pleasure to publish and co-author papers and also Pavel Borodin, Dominik Novotni, Ferenc Kahlesz, Ruwen Schnabel and Markus Schlattmann who have been most pleasant office mates over the years. I would especially like to thank Ralf Sarlette who undoubtedly builds the best BTF measurement devices in the world. Ralf like Gerhard, Jan, Marcin, Fecu, Mirko, Michael, Ákos, Patrick and Roland is also one of the early members of the group (the first generation ☺) and I am quite grateful to them for shaping not only my scientific but also my personal life in many ways I did not even dream of in the beginning.

I will not forget to mention the European Union, the Deutsche Forschungsgemeinschaft and the University of Bonn which were the financing institutions of my work over the years.

And last but by far not least I want to thank my family for their love and endless support - this work is dedicated to You.

CHAPTER 1

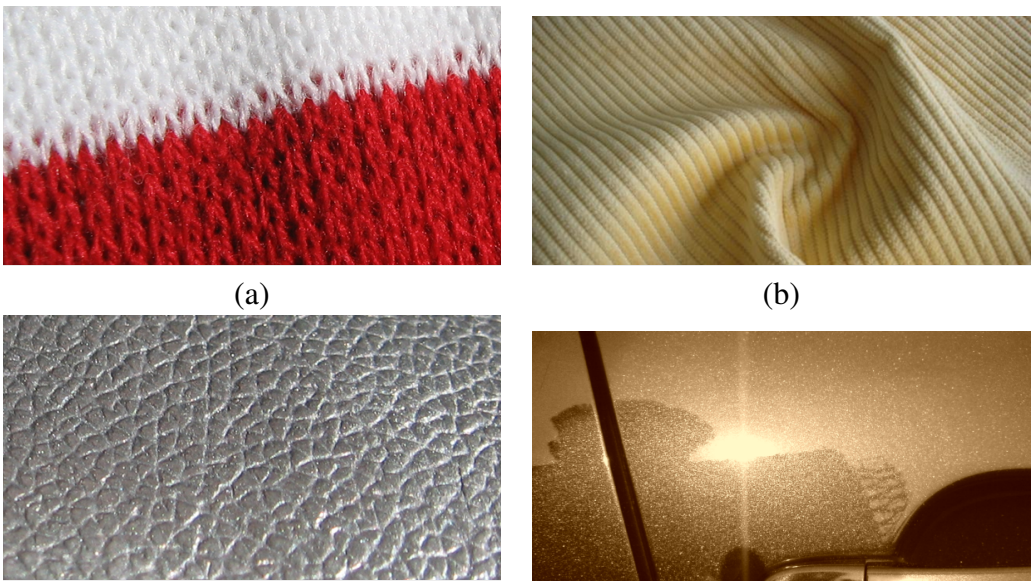
INTRODUCTION

In recent years, data-driven techniques for digital content creation have attracted growing interest both from the computer graphics research community and in practical applications. The simple but powerful idea behind the concept of data-driven representations is to generate new content by intelligently combining elements already available in a database.
Suppose, for example, a digital character needs to be created for a computer game or a digital movie. Classical modeling techniques would require modeling the physical shape of the character using, e.g. subdivision surfaces or NURBS[1] patches. Depending on the detail requested by the application, this may even require having to model every single wrinkle and hair on the skin. If the character also has to be animated, all of these modeled geometric details have to be accurately moved according to the physical and biological laws in order to create a believable animation of the facial expressions.

It is easy to imagine that, even with the quite powerful 3D modeling and animation software available today, such an exercise would require considerable artistic skills and experience, not to speak of the enormous amount of time necessary. Therefore, in current practice, this task is often accomplished rather differently: the physical shape of the character is scanned using a laser scanner. Small geometric detail like wrinkles and hairs are captured in a high-resolution photograph. Subtle facial expressions are recreated by using a motion capture system to record the movements of reflective patches glued to an actors face.

Actually, the striking visual quality and photo-realism of today's high-end computer graphics would not be possible without these kinds of data-driven techniques. In other words, these methods provide solutions for the *complexity and productivity problem* of digital content creation. This trend can be observed not only in the field of computer graphics, but also in other computer science disciplines. The best human speech synthesizers, for instance, are mainly based on recorded samples of real human speech, not on a complete model of the vocal tract.

[1] Non-Uniform Rational B-Spline



(a) (b)

(c) (d)

Figure 1.1: Some materials with complex surface structures at mesoscopic scale (woven and knitted thread structures (a)+(b), scratches and indentations (c), metallic flakes in car paint(d))

It may not be justified yet to speak of a shift of paradigm in modeling of 3D content but it is now commonly accepted that current and upcoming models of reality will one way or the other be based on databases of measurements captured from reality. As a result, the range of required mathematical tools and algorithms increases. Instead of classical parametric and generative models which try to represent reality using a set of parameters and mathematical instructions based on physical assumptions about the world the focus now gears towards methods from statistical data analysis and machine learning which are able to extract and interpret perceptually important properties from a huge data cloud and make them available for effective application in rendering systems.

This thesis offers solutions, mathematical tools and algorithms for the efficient application of data-driven techniques in the field of spatially varying surface appearance, which, in our opinion, is of paramount importance for the creation of believable synthetic images.

1.1 The Complexity of Surface Appearance

When it comes to surface appearance the visual complexity of our real world is particularly evident. For good reason the way of how a surface reflects light is

still surrounded by a kind of magic. Usually, those structures which determine the appearance of materials like metal, plastic or human skin are not visible with the naked eye because the interesting interaction happens at microscopic level. If truth be told, the only scattering phenomena that can be considered to be fully understood are mirror reflection on perfectly flat surfaces, transmissive refraction at perfect refractors and ideal diffuse reflection on perfectly diffuse reflectors. Real surfaces contain irregularities which are usually modeled only statistically "[...], instead of attempting to describe and deal with its surface contours in microscopic detail." [NO77, p.5].

Furthermore, surface appearance is heavily scale dependent: "For example, in examining the reflectance of a highly irregular surface containing deep cavities, [...] it may be possible to consider the reflectance of different portions of the walls of single cavities (which are then regarded as macroscopic irregularities). But when studying the possible effects of similar surfaces which may exist on the moon, [...] these are necessarily treated as microscopic irregularities" [Nic65, p.3].

In classical computer graphics there is a clear distinction between the mentioned macro- and microscopic scales: At macro-scale shape is explicitly modeled using polygons or parametric surfaces while at micro-scale the geometry is modeled statistically using parametric models of reflectance. An interesting aspect is therefore the transition between these scales, i.e. where should we draw the line between explicit and statistical modeling?

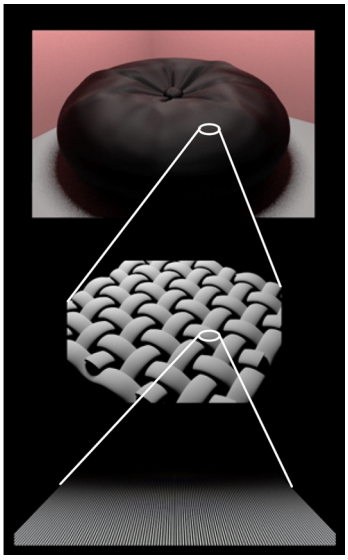


Figure 1.2: Macro-, meso-, and micro-scale in scene modeling. From [WAT92].

Presently, it is common consent in computer graphics that simple blending is not an appropriate solution because it would require the explicit modeling of intricate geometric structures which overburdens rendering systems and causes serious aliasing problems. Therefore, it is convenient to introduce another scale between the macro- and micro-scales which deals with those structures which are too big for statistical modeling (because they are still visible) and too small for explicit geometric modeling (because this would be too expensive). As illustrated in Figure 1.2 this specific scale between macro- and microscopic structures is the *mesoscopic* or just *meso*-scale. Figure 1.1 shows a few materials which exhibit complex structures at this meso-scale. It is not very difficult to imagine that simulating the appearance of such materials using the classic macro- and micro-scale model-

ing paradigms requires serious efforts and that data-driven techniques could be of great value here.

1.2 Image-based Rendering and the BTF

Since the early beginnings of computer graphics texture mapping has been used to improve the realism of computer generated images by adding detail to surfaces [BN76].

Loosely speaking, texture mapping allows to assign a 2D image to a surface in 3D. For example, using texture mapping an image of a textile material can be wrapped around the 3D geometry model of a cloth to add surface details like color patterns and knit structures without modeling them explicitly. This simple approach helped a lot to increase productivity of computer generated imagery because textures can be painted in 2D using paint software or, even better, easily captured from the real world using a digital camera.

This last point deserves special attention. The principle of using images instead of geometric models for scene description and rendering is known as *image-based rendering*. It replaces the traditional modeling and rendering process by a sampling and reconstruction process. The idea is to represent the appearance of an object not by its actual macro-, micro-geometric and photometric properties but by the dense array of emanating rays of light that can be thought of "filling" the space around the object. This array of light rays is known as the *plenoptic function* [AB91] and it is easy to see that it captures the appearance of an object for every possible viewpoint. Thus, taking images of an object using a camera can be considered as sampling the plenoptic function, and reconstructing the continuous plenoptic function from a discrete set of samples can be interpreted as rendering. In this sense, traditional texture mapping which uses only a single color value per texel[2], and was introduced already years before the term image-based rendering was coined, can be interpreted as the reconstruction of the plenoptic function of a material from a single image.

Without going too much into the details of sampling theory, it is obvious that a single image does not suffice to capture the plenoptic function of most materials. In fact, a single image is able to represent only ideal diffuse and flat surfaces which explains the typical cardboard-like look of computer generated images which use only simple textures.

Generally, the appearance of surface points changes if viewed from different directions: highlights move, color and gloss change, occlusion by other surface points occurs, etc. Therefore, an extended texture description which consists of

[2] *texel*=short for ***tex***ture ***el***ement

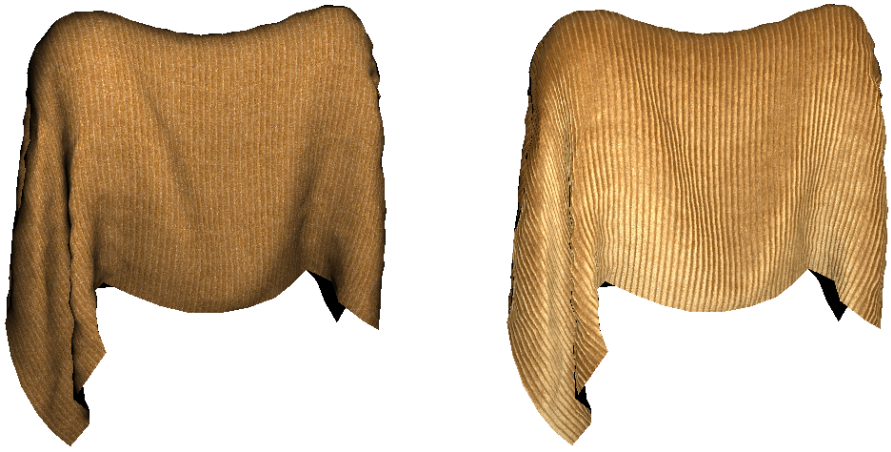


Figure 1.3: Comparing simple texture mapping (left) and BTF-rendering (right).

images from different viewpoints is needed to reconstruct these effects. In the literature this description is known as the *surface light field* [MRP98].

Reconstructing the surface light field of an object from images opened up a new degree of realism but only for completely static scenes. This motivated the introduction of extended image-based representations which can not only be viewed from arbitrary viewpoints but which can be relit, too. The *Reflectance Field* [DHT+00], for example, represents the plenoptic function for varying illumination conditions. Due to the linear nature of light transport new images can be reconstructed (∼rendered) from a sampled reflectance field by linearly combining differently lit samples – a process called *image-based relighting*.

Textures, which can be relit and which reconstruct the surface light field of a material are called *Bidirectional Texture Functions* (BTF). In its original definition, introduced by Dana et al. [DvGNK97] to the Computer Vision community, the BTF is restricted to directional lighting. Figure 1.3 shows a comparison between a simple texture which contains only albedo variation and a BTF which reproduces also the directional appearance variation of the corduroy cloth.

1.3 Main Contributions and Thesis Structure

Image-based rendering and BTFs are now around for at least a whole decade but in comparison to other data-driven techniques like laser scanning or motion capturing they are not quite common in practice (except for standard albedo textures). The main reason for this is the higher dimensionality of image-based representations.

Laser-scanners, for example, capture a cloud of 3D points usually lying on a 2D manifold. In contrast, image-based representations have at least two spatial dimensions and two to four additional angular dimensions representing the variations along view- and light directions. Just to get a glimpse: sampling each of these six dimensions with only 100 samples already results in 1 trillion measurement samples. Thus, the aforementioned complexity and productivity problem of photo-realistic computer graphics has somehow only shifted. Instead of modeling geometric detail and computing expensive light transport we now have to face the following serious challenges:

- **Measurement** - how can appearance be captured efficiently and accurately?
- **Compression** - how can image-based appearance be stored compactly without sacrificing visual fidelity?
- **Rendering** - what is the impact of using representations like BTFs on rendering costs and how can it be minimized?
- **Editing** - how can image-based materials be edited and manipulated without the need to manually touch all the thousands of images?

While practical solutions to these problems are crucial for the applicability of image-based representations like the BTF, exhaustively dealing with all of them would be far beyond the scope of this work. In particular, this thesis will not cover the problem of efficient appearance measurement, which is primarily an engineering problem, neither will it discuss efficient methods for rendering measured materials. In fact, the problem of BTF rendering is strongly coupled with the compression problem since rendering can only be as efficient as the decompression stage of the compression algorithm.

Consequently, this thesis consists of two main parts: **compression** (chapters 3–6) and **editing** (chapters 7-10). In Part I we improve the state-of-the-art in BTF compression by introducing the following scientific and technical contributions:

- **Comparison of matrix factorization techniques** – Many existing data-driven compression techniques make use of matrix factorization. Based on the general model of co-variances, matrix factorization allows to remove redundant information from the data while preserving visually important features. There is an abundance of possibilities how matrix factorization can be applied to high-dimensional data. In Chapter 4 we will compare different matrix factorization methods concerning compression ratio, reconstruction error and reconstruction cost in order to give concrete hints for selecting the most suitable technique for a given situation.

- **Local coordinate systems for image-based rendering** – It is well-known that compression of image-based data generally benefits from known geometry but the extraction and separation of the geometry from the data is difficult. In Chapter 5 we introduce an effective method which exploits approximate geometry information for compression without the need to separate geometry and reflection. The most interesting part of our algorithm is a novel variant of photometric stereo based on the efficient computation of correlations by exploiting the general Fourier-Shift theorem for Spherical Harmonics.

Part II of this thesis introduces one of the very first methods for editing measured BTFs and comprises the following contributions:

- **BTF Analogies** – The sheer size of image-based representations makes them particularly difficult to edit. Methods are required which propagate simple editing operations to the high-dimensional representation. In Chapter 8 we present our BTF Analogies technique which generalizes the well-known Image Analogies framework [HJO+01] to BTFs. The algorithm is able to interactively and consistently propagate edits applied to proxy images to the whole data.

- **Meaningful BTF Interpolation** – Based on the BTF Analogies we propose a method for meaningful interpolation of BTFs. By employing also a procedural texture model new BTFs can be created from a database of measurements in a meaningful and intuitive way (Chapter 9).

Beyond these two main parts the thesis contains furthermore a background chapter which follows immediately afterwards and presents the necessary preliminaries of the theory and practice of image-based material representations like the BTF. It includes an introduction into the theoretical foundations of surface reflectance and light transport. The thesis is concluded in Chapter 11 where also possible directions for future research will be discussed.

As usual in the field of computer graphics most of the algorithms presented in this thesis have already been published at several international computer graphics conferences [MMK03, MMK04b, MBK05, MSK06, MSK07]. We also presented several of the techniques discussed here in a well-recognized state-of-the-art report [MMS+05] and at various tutorials at the annual Siggraph [LGC+05] and Eurographics [LGM07] conferences.

CHAPTER 2

PRELIMINARIES OF VISUAL APPEARANCE

The *visual appearance* of an object refers to everything that can be perceived of it with the naked eye. This includes geometric attributes like shape and size but also material attributes like color or translucency as well as more fuzzy properties like softness, brittleness, age, etc. The simulation of such material appearance attributes with a computer is particularly challenging, because the geometric structures involved are usually quite small which makes an accurate physical simulation infeasible. Therefore, reasonable abstractions have to be used which can be based on physics but above all should condense the physical detail into compact models which can efficiently be evaluated by computers.

The most fundamental abstraction in Computer Graphics is probably the light transport- or rendering equation. Since it is elaborately covered in most standard Computer Graphics textbooks (e.g. [CWH93] or [PH04]) we will give only a brief introduction to the rendering equation and the necessary radiometric terms in Sections 2.1 and 2.2.

The main concern of this thesis is the visual appearance of materials. Therefore, we will state more precisely what materials are and how they are categorized for the purpose of visualization in Section 2.3. Then we will introduce the two different paradigms current material models are based on: at first physically-based, parametric modeling of appearance (Section 2.4) and, secondly, image-based measurement and rendering (sections 2.5 and 2.6).

The central definition of this chapter is the light transport matrix (Equation 2.14). Our (simplified) derivation was inspired by Lehtinen's framework for precomputed and captured light transport [Leh07] which is strongly recommended for further reading.

2.1 Basic Radiometry

Computer Graphics is a very young scientific discipline. Nevertheless, its theoretical foundations range back more than 3000 years into the classic discipline of optics. Optics is traditionally studied on three different scales where each scale has its own strengths and weaknesses in explaining specific visual phenomena.

On the scale of single atoms *quantum optics* describes the interaction between photons and molecules or even atoms which is usually too detailed for the purpose of visualization. *Wave optics* is concerned with the interaction of light with objects the size of which lies in the range of the wavelength of light. It enables to describe effects such as polarization and diffraction which cause, e.g. the familiar rainbow patterns that appear on a compact disc. However, these effects are usually not modeled explicitly in Computer Graphics.

Instead, most image generation algorithms are based on the principles of *ray optics* which explains the properties of light on a macroscopic scale. It is assumed that light travels instantaneously through a medium along straight lines. An important consequence of ray optics is that the light distribution at a point $\mathbf{x}$ in space is given by the rays $L_i(\mathbf{x}, \omega_i)$ incident at $\mathbf{x}$ from each direction $\omega_i \in \Omega$. In order to quantify such a light distribution we need to specify its physical units. In particular, we need to specify the units of energy carried by a single light ray. Since in physics light is thought of as electromagnetic radiation the units of light are usually described in terms of radiometry which is the science of the physical measurement of electromagnetic energy. The important unit for Computer Graphics which quantifies the amount of energy carried by a single ray is *radiance*. It will be derived in the following subsection.

2.1.1 Radiometric Quantities

The fundamental quantity in radiometry is the radiant energy Q expressed in $Joule[J]$. Intuitively, an amount of light energy corresponds to a number of photons where one photon has an energy of $\hbar \cdot \frac{c}{\lambda}$ and where $\hbar$ is Planck's constant, c is the speed of light and λ the wavelength associated with the photon. Since counting photons traveling with the speed of light can be rather tedious (even for computers ☺), differential quantities are much more convenient in specifying light energy.

The radiant power $\Phi = \frac{dQ}{dt}$, also called *flux*, which is expressed in *Watt* $[W] = [Js^{-1}]$, denotes the number of photons (which means the amount of energy) flowing from/to/through a surface per unit time.

Power does not give any information about the density of the light flow. This brings us to *irradiance* $E = \frac{d\Phi}{dA}$ which is expressed in $[Wm^{-2}]$ and quantifies the incident power per unit surface area. The emitted analogue to irradiance is

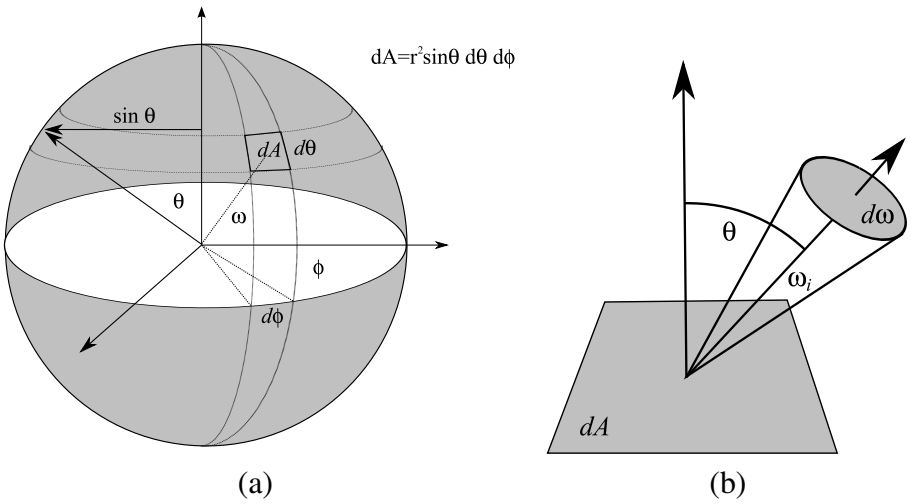


Figure 2.1: (a) Geometry of the differential solid angle. By definition the solid angle subtended by a spherical area A is equal to A/r^2. It follows $d\omega = dA/r^2 = \sin\theta d\theta d\phi$. (b) Geometry of radiance: flux per differential projected area $dA^\perp$ per differential solid angle $d\omega$ in the direction ω_i.

radiosity which is a well-known quantity in Computer Graphics since the radiosity algorithm, which essentially computes the radiosity of diffuse surface patches, has been named after it.

Knowing the density of the photon flow at a given point does not suffice to describe the appearance of the point when viewed from a particular direction. Intuitively, we need to specify the photon density per direction. Since a direction has zero extent we also need to define something like a "direction with extent" which is the role of the differential solid angle $d\omega$ illustrated in Figures 2.1.

In terms of flux we then have the radiance $L(\mathbf{X}, \omega_i) = \frac{d^2\Phi}{d\omega dA^\perp}$ (expressed in $[Wsr^{-1}m^{-2}]$) where $dA^\perp = dA\cos\theta$ is the projected area of the differential surface element dA onto the hypothetical differential surface perpendicular to ω_i which subtends the solid angle $d\omega$ (see Figure 2.1 (b)). The cosine term stems from the fact that the emitted power is "smeared" over a larger area when seen from shallow angles.

Radiance is the fundamental radiometric quantity in ray tracing algorithms since it remains constant along rays in empty space and thus describes the energy "carried" by a ray. Furthermore, as indicated above, rendering a scene from a given view point can now be interpreted as computing the radiance incident at the view point. Last but not least, other quantities like irradiance and total flux can be computed by integrating over directions and area respectively. Light sensors such as cameras and the human eye are sensitive to radiance in the sense that their

Physics	Radiometry	Photometry
Flux	Radiant power $[W] = [Js^{-1}]$	Luminous power (lumens)
Flux density	Irradiance $[Wm^{-2}]$	Illuminance (lux)
Angular flux density	Radiance $[Wsr^{-1}m^{-2}]$	Luminance (nit)

Table 2.1: Radio- and photometric units

response is always proportional to the incident radiance.

2.1.2 Photometric Quantities

Since the actual response of the human eye to light energy is not equal across the visible spectrum it is useful to know also about the relations between radiometry and photometry which deals with the perception of light energy in the human brain. Historically, radiometry evolved from the area of photometry but since the radiometric terms are more general than photometric terms and the photometric quantities can be computed from radiometric quantities it is common standard to use radiometric quantities in Computer Graphics.

The radiometric quantities introduced above can be derived purely from the geometry of flows (which we omitted here, see e.g. [CWH93, chp.2]) but important properties of radiation such as the inverse square law have already been discovered before the founding of radiometry while studying human vision. The classical units from photometry can be derived from the general radiometric units by integrating them against the so-called *spectral luminous efficiency function* $V(\lambda)$ which quantifies the sensitivity of the human eye for a given wavelength λ and is defined for daylight (photopic) and night vision (scotopic). $V(\lambda)$ is greater than zero for wavelengths between 380 and 780 nm and has its maximum around 550 nm which is perceived as green.

Table 2.1 shows an overview of the relevant photometric units and their radiometric counterparts.

2.2 The Rendering Equation

The formal framework of rendering is the *rendering equation* as introduced by Kajiya [Kaj86]. It is based on geometric optics and the law of conservation of energy. We give the directional formulation for opaque surfaces here:

$$L_o(\mathbf{x}, \omega_o) = L_e(\mathbf{x}, \omega_o) + \int_{\Omega_i^+} \hat{f}(\mathbf{x}, \omega_i, \omega_o) L_i(\mathbf{x}, \omega_i) \cos\theta_i \, d\omega_i \qquad (2.1)$$

Accordingly, this equation describes the outgoing radiance L_o at a point $\mathbf{x}$ on a surface in direction ω_o as the sum of the emitted radiance and the reflected radiance which is the integral over the upper hemisphere of incoming radiance L_i times the spatially varying material properties $\hat{f}$ (which will be introduced in Section 2.4) times the foreshortening factor $\cos\theta_i$. Since the incoming radiance $L_i(\mathbf{x}, \omega_i)$ can be interpreted as the outgoing radiance at the point seen from $\mathbf{x}$ in direction ω_i the rendering equation consists of a recursive and thus infinite-dimensional integral. Mathematically, equations of this form are called *Fredholm equations of the second kind* which in general have no known analytic solution. Therefore, rendering algorithms rely on numerical techniques like Monte-Carlo integration, which is often implemented by ray-tracing algorithms, to find approximate solutions for L_o (see, e.g. [PH04] for an introduction). Please note, that the rendering equation is associated with surface points $\mathbf{x}$ which have local reflectance properties $\hat{f}(\mathbf{x})$ which means that non-local effects like shadows or inter-reflections have to be computed as part of solving the integral equation. Furthermore, the equation assumes opaque materials, i.e. light is directly reflected at the point of incidence.

In order to simulate also objects with translucent materials the definition of surface scattering behavior and the domain of integration has to be extended. In particular, we need to integrate over incoming directions *and* surface area of the translucent object:

$$L_o(\mathbf{x}, \omega_o) = L_e(\mathbf{x}, \omega_o) + \int_A \int_{\Omega_i^+(\mathbf{x}_i)} S(\mathbf{x}_i, \omega_i; \mathbf{x}, \omega_o) L_i(\mathbf{x}_i, \omega_i) \cos\theta_i d\omega_i dA(\mathbf{x}_i) \quad (2.2)$$

This way, also light incident at point $\mathbf{x}_i \neq \mathbf{x}$ and scattered to $\mathbf{x}$ is integrated. The scattering behavior of the material is encoded in the general scattering function S.

2.3 Material Classification for Computer Graphics

Material in the sense of *stuff* is a *substance that goes into the makeup of a physical object*. A wall, for example, is made of stone, a cup is made of porcelain, etc. Human language has developed a rich vocabulary for describing and categorizing such stuff but it remains unclear how most of these categories translate into formal representations of materials. For instance, how a "*fluffy*" material should be described mathematically is still an open question. Therefore, in Computer Graphics materials are usually classified based on the complexity of their mathematical representation which then constrains the visual phenomena that can be represented by it. For example, in Equation 2.1 materials are always *opaque* because the incoming light is only integrated over the upper hemisphere. Firstly, let us specify more rigourously what parts of a scene are defined as material in Computer Graphics. Not surprisingly, this is a question of scale.

2.3.1 Hierarchy of Detail

Every visual surface characteristic is the result of an interaction between light and matter. As already mentioned, the explicit modeling of all geometric details is infeasible and therefore, not without reason, the rendering equation separates between explicitly modeled geometry on the one hand and geometry that is encoded within the "black-box" scattering functions $\hat{f}$ and S on the other hand. This separation can be understood as two different levels of detail which demand different representations. These levels are not static but depend on scale, e.g. the surface of the moon can be regarded as a material when observed from the earth but it should be modeled as geometry for a virtual astronaut landing on it. In fact, we can imagine an infinite hierarchy of levels of detail $\ldots D_{i-1} \subset D_i \subset D_{i+1} \subset \ldots$ starting from intergalactic dimensions down to single atoms and so on[1]. In order to deal with this vast complexity Kajiya [Kaj85] suggested to consider three different scales at each particular level of detail D_i (see also Figure 1.2):

- The *macroscopic* scale contains the geometry which can be modeled effectively using surface models (like polygons or polynomial patches). This scale is usually associated with the shape of objects.

- The *surface mapping*- or *mesoscopic*-scale contains details which would be geometrically modeled at level D_{i+1} and are usually represented using textures and bump- or displacement maps which are parameterized across the geometric model.

- Even smaller details which could not be resolved by the display device are from the *microscopic* scale. Representing details from this scale using surface models or maps would result in excessive aliasing. Therefore, the influence of these details on appearance should be accumulated into reflection models, i.e. into the functions $\hat{f}$ and S from equations (2.1) and (2.2).

The meso- and microscopic scales are usually associated with the material an object is made of, i.e. with stuff. The geometry present at these scales gives objects their *characteristic look*. For example, the unique appearance of a corduroy cloth is defined by the weaving patterns (meso-scale) and the reflectance properties of individual threads and groups of threads (micro-scale).

This hierarchy of detail suggests a first fundamental classification of materials: *Homogenous* materials which are solely defined by their micro-structure, or, equivalently, by their *directional appearance* expressed by the change in appearance when the viewing position or the lighting changes, and *heterogenous*

[1] An impressive movie illustrating this infinite hierarchy of detail can be found at `http://www.powersof10.com`

materials that are additionally characterized by spatial variations on mesoscopic scale.

2.3.2 Homogenous Materials

Homogenous materials can be further classified based on their opacity.

***Opaque* materials** mainly reflect light which means that the in-scattered fraction of light can be neglected. How the light is reflected depends on the micro-structure which can be roughly characterized as follows. A very *rough* (stochastic) micro-structure scatters light uniformly in all directions which results in *diffuse* reflection. The other extreme is a perfectly flat surface which reflects light only into the *mirror* direction (as determined by the law of reflection). In between are *glossy* surfaces which spread the reflected light around a preferred direction (usually the mirror direction). The width of this so-called scattering *lobe* correlates with the roughness of the micro-structure. For many materials this lobe is *isotropic* which means that the material does not change its appearance when rotated around the surface normal. On the opposite, if the micro-structure has a preferred alignment direction, e.g. is made of oriented fibers or grooves, the reflection becomes *anisotropic* because then it depends on the (projected) angle between the light beam and the alignment direction of the material.
***Translucent* materials** show considerable light scattering within the material. Many translucent materials can be characterized statistically by the distribution of light absorbing or -scattering particles. Optically thick materials can often be regarded as diffuse materials because the scattering from particles resembles scattering from rough surfaces. If the optical thickness decreases the resulting subsurface scattering creates a distinct look known from fluids like milk or natural materials like leaves or marble.
***Transparent* materials** let the light get through without any scattering. Transparent materials are fully characterized by their index of refraction which determines how the light ray is refracted when it enters or leaves the material.

Many real materials are combinations of the above classes since they might be made out of different layers or show a considerable amount of reflection *and* in-scattering. Often, the fraction of reflected and absorbed or in-scattered light depends on wavelength which causes the sensation of a colored material. Many heterogenous materials are made of spatial patterns of homogenous materials.

2.3.3 Heterogenous Materials

In reality almost all materials show some heterogeneity due to imperfections, dirt, dust, etc. However, a classification of heterogenous materials is difficult and exist-

ing ones are often inappropriate for modeling in Computer Graphics. Therefore, spatial variation is usually represented explicitly (painted or captured) in 2D using textures and mapped to macro-scale geometry. Nevertheless, there is a very important distinction that can be made between heterogenous materials, namely the complexity of their surface height variation on mesoscopic scale.

***Flat* materials** can be fully characterized by spatially varying micro-scale geometry, i.e. they can be interpreted as spatial patterns of homogenous materials. Examples are printed paper or fabrics with microscopically small weaving patterns.
***Bumpy* materials** have visible surface height variations which cause shading, parallax, shadows, and masking effects. The height variations can be represented by a 2D heightfield (displacement map).
***Volumetric* materials** cannot be assigned a single depth value per surface point. Besides shadowing and masking effects these materials usually show complex light scattering like inter-reflections and sub-surface scattering. Examples are shag pile or natural surfaces like grass or fur.

Another type of classification is to consider the statistical distribution of textural features like color and contrast gradient. The key element here is that most heterogenous materials are made of repeating elements or patterns which are distributed in a certain way. If these distributions are known they can be used to generate arbitrary large instances of the material.

***Stochastic* materials** have a more or less random distribution of patterns. Materials like sand or ingrain wallpaper are of this type. Based on the alignment of patterns we can distinguish between isotropic materials, which have no preferred alignment direction, and anisotropic materials.
***Semi-regular* materials** consist of repeating structures where each repetition introduces slight variations in color, geometry, etc. For example, materials resulting from growth processes like lizard skin are semi-regular.
***Regular* materials** are made of strictly repeating structures. Such materials are usually created by a regular or automatic process like textiles made from synthetic fibers, checkerboard or bathroom tiles etc.

Materials that contain also global patterns like a few spatially varying scratches on a lacquered surface can be termed *globally variant* textures [WHZ+08].

2.4 Classical Reflectance Modeling

The theory of reflectance modeling provides the basic geometry and nomenclature for the reflection functions used in the formulation of the rendering Equations

(2.1) and (2.2). Classical reflectance modeling usually refers to the micro-scale aspects of scene modeling by abstracting from any visible shape and surface details.

2.4.1 Reflectance on Idealized Surface Patches

We assume that a beam of light is incident on an idealized, locally flat surface patch at point $\mathbf{x}_i$. Neglecting the dependency on time and assuming a fixed wavelength band λ_0 for both incoming and reflected rays (that means ignoring effects like *phosphorescence* and *fluorescence*) the appearance of a point $\mathbf{x}_o$ on this surface patch now only depends on the incident light beam $(\mathbf{x}_i, \omega_i)$ and the amount of energy that is reflected from $\mathbf{x}_o$ in the direction ω_o of the observer. In general, the amount of reflected energy is proportional to the incoming flux. The corresponding 8-dimensional proportionality factor is termed *bidirectional scattering-surface reflectance distribution function]* (BSSRDF) [NO77]:

$$S_{\lambda_0}(\mathbf{x}_i, \omega_i; \mathbf{x}_o, \omega_o) := \frac{dL_{r,\lambda_0}(\mathbf{x}_o, \omega_o)}{d\Phi_{i,\lambda_0}(\mathbf{x}_i, \omega_i)} \qquad [sr^{-1}m^{-1}] \tag{2.3}$$

The BSSRDF alone is only a mathematical abstraction, a "*black-box*". Quoting Nicodemus, it "*merely provides a way of quantitatively expressing the connection between reflected flux leaving $\mathbf{x}_o$ in a given direction and the flux incident at $\mathbf{x}_i$ from another given direction [...] which produces it*" [NO77, p. 4]. Nothing is said about the actual mechanism involved or any material existing in reality and the representation is completely decoupled from the actual (micro-)geometry. To this end the BSSRDF provides only a formalization of the basic reflection geometry.

The BSSRDF is part of Equation (2.2) and the main task of appearance modeling in Computer Graphics is to find computable models based on the BSSRDF reflection geometry which describe the reflectance behavior of real materials and can be used in rendering systems. This will relieve the modeling system from explicitly representing the geometry up to micro-scale and computing a full light transport simulation for this geometry. The question remains, how these practical BSSRDF models can be found.

The first step is usually to restrict oneself to one of the already mentioned subclasses of materials. A common restriction is to assume a homogenous and opaque material because then subsurface scattering can be neglected and we have uniform reflection properties across the whole surface patch. In this case the dependency on spatial position can be dropped for the in- *and* outgoing ray and we arrive at the famous 4D *bidirectional reflectance distribution function* (BRDF):

$$f(\omega_i, \omega_o) := \frac{dL_o(\omega_o)}{dE_i(\omega_i)} \qquad [sr^{-1}] \tag{2.4}$$

which is the ratio of differential outgoing radiance to incoming irradiance. Note, that the spectral index λ_0 is usually omitted for brevity. In contrast to the BSSRDF, finding appropriate models for the 4D BRDF is easier and many different models have been published over the years.

2.4.2 BRDF Models

Real BRDFs are strongly constrained 4D functions. They have to fulfill physical constraints like *Helmholtz reciprocity* and *energy conservation*. Furthermore, as sketched in Section 2.3, the appearance of many homogenous materials is more or less defined by the statistical distribution of orientations and slopes of the underlying micro-structure and for many materials an *isotropic* distribution can be assumed. The BRDF models developed based on these kinds of observations can roughly be categorized into empirical models and physically-based models. The first group consists of models that where designed to mimic the typical reflection behavior of real materials "*paying little attention to the physical derivation of the model, or the physical significance of its parameters*" [War92, p.1]. Instead, a physically based model "*attempts to get closer to the true distribution by starting from physical theory*"[War92, p.1].

The most simple BRDF model is assuming an ideal diffuse or *lambertian* reflector. In this case the BRDF is simply a constant:

$$f(\omega_i, \omega_o) = \frac{k_d}{\pi} \tag{2.5}$$

with $0 \leq k_d \leq 1$ which assures energy conservation. While a lambertian BRDF is physically plausible (it fulfills the necessary physical constraints) and a reasonable approximation for materials like chalk or paper no existing surface is ideal diffuse.

For modeling glossy reflection several approaches have been been published in the literature. We introduce only models relevant for this thesis here and refer the interested reader for more thorough overviews to [PH04, Chp. 9] or [NDM05].

Cook-Torrance Model The model introduced in [CT81] assumes that the micro-geometry of the surface consists of randomly distributed and oriented specular microfacets. The formula is

$$f(\omega_i, \omega_o) = \frac{k_d}{\pi} + \sum_{j=0}^{K} \frac{k_{s_j}}{\pi} \frac{F_{R_{0,j}}(\gamma_h) \cdot D_{m_j}(\theta_h) \cdot G(\omega_i, \omega_o)}{\cos\theta_i \cdot \cos\theta_o} \tag{2.6}$$

and it consists of an ideal diffuse term which accounts for view-independent internal and multiple scattering and a multi-lobe glossy term. Each lobe consists of a Fresnel term F (usually Schlick's approximation is used here [Sch94]) with

parameter $R_{0,j}$, the microfacet distribution D which by default is the Beckman distribution with parameter m_j, and the shadowing term G. The number of glossy lobes K is equal to 1 in the original paper, multiple lobes can be used for modeling layered materials.

Lafortune Model The famous Phong reflectance model [Pho75] is the typical example for an empirical model since it just mimics the blurred light-source reflections (highlights), taking place on materials like opaque plastic with a cosine function raised to a power. Driven by its several deficiencies like physical implausibility and restriction to isotropic materials Lafortune et al. [LFTG97] designed a generalized version which has the following form:

$$f(\omega_i, \omega_o) = \sum_{j=0}^{K} k_{s_j}(C_{x,j}\omega_{i,x}\omega_{r,x} + C_{y,j}\omega_{i,y}\omega_{r,y} + C_{z,j}\omega_{i,z}\omega_{r,z})^{m_j} \tag{2.7}$$

where $(\omega_x, \omega_y, \omega_z)^t$ is a direction in Cartesian coordinates and k_{s_j}, $C_{x,j}$, $C_{y,j}$, $C_{z,j}$ and m_j are the per-lobe parameters. The lobes can be designed to model various effects like non-lambertian diffuse reflection ($C_x = C_y = 0$), retro-reflection ($C_x, C_y, C_z > 0$) and also anisotropic reflection ($C_x \neq C_y$). The model is popular in the graphics community because it can be evaluated efficiently even on older graphics hardware and can be fitted to measured data.

2.4.3 Beyond BRDF-Modeling

Subsurface Scattering

For a relatively long time the accurate simulation of translucent materials was only possible by solving the general scattering equation for participating media or, as it is usually referred to in Computer Graphics, the volume rendering equation:

$$(\omega \cdot \nabla)L(\mathbf{x}, \omega) = -\sigma_t L(\mathbf{x}, \omega) + \sigma_s \int_{\Omega} p(\mathbf{x}, \omega, \omega')L(\mathbf{x}, \omega')d\omega' + \epsilon(\mathbf{x}, \omega). \tag{2.8}$$

This is an integro-differential equation and it describes the difference in radiance within the medium along direction ω due to absorbtion and out-scattering (given by extinction coefficient σ_t), in-scattering (determined by scattering coefficient σ_s and the phase function p) and emission (quantified by the source term ϵ). Similarly to the standard rendering equation (2.1) this equation can also be solved using Monte-Carlo techniques like ray-marching but it is easy to imagine that this is quite expensive in the general case.

Therefore, rendering of translucent materials became practical for the first time with the introduction of the first parametric models for translucent materials. They were based on assuming a homogenous and highly scattering material. In this case the outgoing light distribution becomes isotropic, i.e. only depends on the distance $|\mathbf{x}_i - \mathbf{x}_o|$, and can be computed by simulating [Sta95] or modeling [JMLH01] a diffusion process. The BSSRDF model for highly scattering materials as introduced in [JMLH01] reads as follows:

$$S_d(\mathbf{x}_i, \omega_i, \mathbf{x}_o, \omega_o) = \frac{1}{\pi} F_t(\mathbf{x}_i, \omega_i) R_d(|\mathbf{x}_i - \mathbf{x}_o|) F_t(\mathbf{x}_o, \omega_o) \tag{2.9}$$

where R_d is the radial diffuse subsurface scattering function which is modeled using a diffusion dipole depending on the absorbtion and scattering coefficients of the material (see [JMLH01] for details). Materials whose appearance is dominated by low order scattering events are usually simulated directly and finding efficient analytical models for this case is still an open research problem.

Spatial Variation

As mentioned above, real materials are hardly perfectly homogenous. If the material is heterogenous but opaque then sub-surface scattering can be neglected and we have a spatially varying BRDF (SVBRDF) (we are still assuming a flat surface patch):

$$\hat{f}(\mathbf{x}, \omega_i, \omega_o) := \frac{dL_o(\mathbf{x}, \omega_o)}{dE_i(\mathbf{x}, \omega_i)} \qquad [sr^{-1}] \tag{2.10}$$

The SVBRDF represents the spatially varying reflectance in the rendering equation (2.1) and is usually represented by discrete 2D coefficient maps (textures) which spatially modulate the parameters of BRDF models. These texture maps can be painted, procedurally generated, synthesized from examples or measured from real objects (see Section 2.5).

Mesoscopic Surface Details

All models presented so far only represent the appearance resulting from (heterogenous) micro-scale geometry. In order to avoid the explicit geometric modeling of mesoscopic surface details *normal-* and *displacement mapping* have been developed (see [SKU08] for an extensive overview).

Normal maps simulate the shading patterns resulting from mesoscopic geometry [Bli78]. The idea is to compute the shading (evaluation of the BRDF) using the normal vectors from the mesoscopic geometry instead of the macro-scale surface normal vectors. Since no real geometry is created they do not reproduce

3D effects like parallax, shadowing and masking and as a result normal mapped surfaces can appear synthetically flat.

Displacement maps in their original form [Coo84] generate real geometry by displacing mesh vertices in the direction of their normal vector which naturally reproduces correct parallax and occlusion. Novel pixel-based methods either perform some sort of ray-tracing on the GPU (e.g. [POJ05]) or store pre-computed displacements (e.g. [WTL+04]) which are accessed during rendering in order to displace the texture access. The latter approach can also be used to render non-heightfield meso-geometry.

Both mapping methods provide efficient methods for the representation and rendering of mesoscopic surface details and often support efficient self-shadowing but they do not solve the problem of efficient simulation of complex higher-order light-transport on these materials.

2.5 Image-Based Rendering and the BTF

As sketched in the previous section, Computer Graphics possesses a rich bouquet of models and representations of material appearance. Based on these models and global illumination algorithms synthetic images with photo-realistic material appearance can be created with a computer. However, practically there is still a long way to go until the first convincing rendering which reproduces the favorite skirt-fabric or the beloved leather wingback chair will yield a satisfying result. As Paul Debevec put it in the introduction of the now classical Siggraph course on image-based modeling and rendering [DG98]: "*modeling is hard, and rendering is slow*".

These restrictions of *geometry based rendering* drove the development of *image-based rendering* and modeling techniques which aim at capturing geometric detail and appearance from photographs. While the term "image-based rendering" was coined as recently as in the 1990s, rendering techniques based on images like texture-mapping [BN76] where introduced already in the pioneering years of realistic rendering. These image-based techniques should not be seen in competition to rigorous measurement of reflectance which remains the area of expertise of physicists (e.g. [PB96]) but they offer solutions for the efficient generation of images with photo-realistic appearance.

The theoretical framework of image-based rendering and relighting is the reflectance field which is based on the plenoptic function.

2.5.1 Light Fields and the Plenoptic Function

Photo-realistic image generation can be interpreted as computing a set of light rays which create the sensation of a realistic scene in the observer's eye. This set of light rays will be a subset of of all light rays in the scene for a fixed moment in time. All light rays and their associated radiances are represented by the *plenoptic function* $P(\mathbf{X}, \omega)$ [AB91] which assigns radiance to all rays originating from every point in space.

Realizing that single images of a scene are 2D slices of the plenoptic function and that it can be reconstructed from a set of these slices via appropriate interpolation was the major insight of Gortler et al. [GGSC96] and Levoy & Hanrahan [LH96] and kicked-off image-based rendering. In short: while rendering based on approximate, iterative solutions of the rendering equation computes values of the plenoptic function for a given viewpoint, image-based rendering reconstructs the plenoptic function from a set of samples.

For rendering purposes it is often convenient to enclose the whole scene/object whose plenoptic function is about to be sampled in a bounding volume V. Then it is assumed that the viewer is always located outside the bounding volume. It results that the rays originating from the bounding volume surface are sufficient to represent the complete plenoptic function of the object (in vacuum radiance does not change along a ray). In computer graphics this simplified 4D-version of the plenoptic function is called the (outgoing) *light field* $L_{o,V}(\mathbf{x}, \omega)$.

Please note, that it is also possible to place the viewer *inside* the bounding volume. This way, the radiance entering the volume is stored which allows to render for example walk-through animations. By convention these light fields are called *incident light fields* $L_{i,V}(\mathbf{x}, \omega)$.

2.5.2 The Reflectance Field

Light field rendering allows to virtually inspect real objects and whole scenes with previously unmatched realism. Honestly, doing image-based rendering this way has more similarities with photography than with rendering since apart from the variable viewpoint these renderings are completely static. In fact, image-based rendering is performed without explicit knowledge of geometry and materials and hence it is not clear how these parts of the scene could be altered.

Due to the linearity of light transport this problem can be alleviated in case of varying illumination: Adding the plenoptic functions of a scene lit by two different illumination conditions will yield the plenoptic function of the scene as being lit by the combination of the two illuminations. In short (the incoming illumination is indicated as subscript):

$$P_{L_1+L_2} = P_{L_1} + P_{L_2}.$$

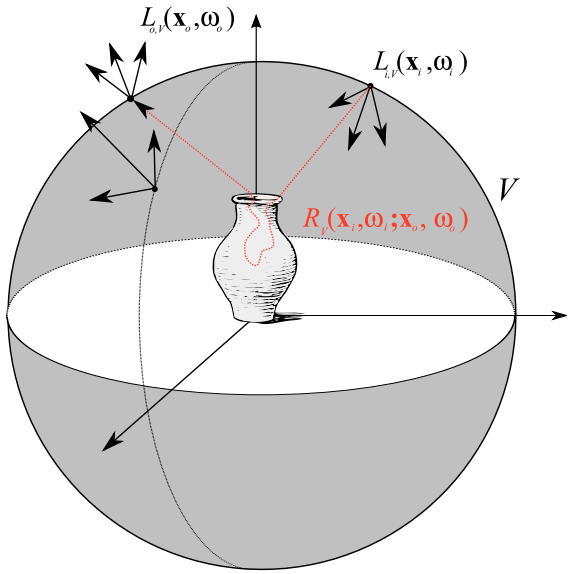


Figure 2.2: The reflectance field $\mathcal{R}_V$ transfers every incident light field $L_i(\mathbf{x}_i, \omega_i)$ to its corresponding outgoing light field $L_{o,V}(\mathbf{x}_o, \omega_o)$ parameterized on bounding surface V. The scene itself can be arbitrarily complex.

Let us now assume again that the whole scene is enclosed within a bounding surface V (see Figure 2.2). Let us furthermore assume that the illumination only originates from outside the volume (if the scene emits light this light will remain static). Then the incoming illumination can completely be represented by an incident light field $L_{i,V}(\mathbf{x}, \omega)$ parameterized on the surface V of the bounding volume. Based on the linearity of light transport it would now be possible to compute the outgoing light field $L_{o,V}(\mathbf{x}, \omega)$ for an arbitrary incident light field if we had some kind of transport function $\mathcal{R}_V$ that encodes the outgoing light field for every single incident ray $(\mathbf{x}_i, \omega_i)$ using the following integral:

$$L_{o,V}(\mathbf{x}_o, \omega_o) = \int_V \int_{\Omega_i^+(\mathbf{x}_i)} \mathcal{R}_V(\mathbf{x}_i, \omega_i; \mathbf{x}_o, \omega_o) L_{i,V}(\mathbf{x}_i, \omega_i) d\omega_i d\mathbf{x}_i \tag{2.11}$$

Since the incident light field is defined outside of the bounding volume of the scene this integral is not recursive in contrast to the rendering equations (2.1) and (2.2). We might call equation (2.11) the *image-based relighting equation*. The transport function $\mathcal{R}_V$ is defined as the *reflectance field* [DHT+00] and it represents the outgoing light field of the scene for every possible incoming illumination. $\mathcal{R}_V$ is principally equivalent to the BSSRDF S (Equation (2.3)) but it is not asso-

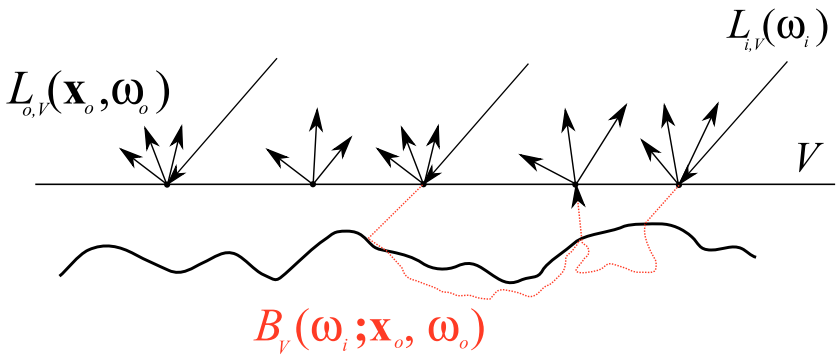


Figure 2.3: The BTF $\mathcal{B}_V$ transfers incoming directional lighting $L^d_{i,V}(\omega_i)$ to the corresponding outgoing light field $L_{o,V}(\mathbf{x}_o, \omega_o)$ parameterized on a planar reference surface V. In contrast to a general *far-field reflectance field* the scene is usually an approximately flat surface but with arbitrarily complex mesoscopic geometry.

ciated with a physical surface. Instead, it is associated with the bounding surface V which can enclose arbitrarily complex scenes with arbitrarily complex geometry and light transport. In other words, while the BSSRDF is meant to encode light transport on a surface the reflectance field can represent the appearance of complex scenes in a black-box manner.

2.5.3 Bidirectional Texture Function

The term Bidirectional Texture Function (BTF) was already known in the field of Computer Vision before it was introduced to the graphics community by Dischler [Dis98] as "*a more general formulation of the texture mapping principle*".

It is easy to confuse the BTF with a SVBRDF (Equation 2.10) which probably would be a more convenient name for spatially varying appearance because the term BRDF is so well-known but there are subtle differences. First, we have the same distinction as between reflectance field and BSSRDF: for BTFs the geometry of the texture does not need to coincide with the parameterized surface. Therefore, the BTF can contain real visible geometry with parallax and masking effects which are not part of the SVBRDF concept. Second, the BTF is assumed to be lit by directional light which means that surface points are not only hit by direct light but also by light scattering from neighboring surface points which contradicts the definition of the SVBRDF, too.

Consequently, we propose to interpret the BTF as a 6-dimensional reflectance field $\mathcal{B}_V$ which is parameterized on a (typically planar) surface V and encodes

light transport for *directional* incident light fields which do not vary with surface position $\mathbf{x}_i$ (far-field illumination). Therefore, given the BTF $\mathcal{B}_V$ of a scene its outgoing light field L_o can be computed for an arbitrary, directional, incident light field $L^d_{i,V}$ as follows:

$$L_{o,V}(\mathbf{x}_o, \omega_o) = \int_{\Omega_i^+} \mathcal{B}_V(\omega_i, \mathbf{x}_o, \omega_o) L^d_{i,V}(\omega_i) d\omega_i \tag{2.12}$$

The restriction to far-field illumination cuts two dimensions from the full reflectance field which means that the BTF is a more lightweight representation compared to a general reflectance field. On the downside light transport between different surface points cannot be resolved. Keeping in mind that the typical application of BTFs are materials with significant meso-scale structures this is not a huge restriction because most of these materials are optically thick and compared to the size of the mesoscopic structures the incoming light is usually slowly varying. Of course, there will be artifacts if the material is lit by laser-pointer-like illumination or at sharp shadow boundaries but on the other hand the complex spatially varying structure of textures often hides these kinds of artifacts.

2.6 Image-Based Appearance Acquisition

2.6.1 Light Transport Matrix

In sections 2.4 and 2.5 the BSSRDF and its image-based pendant, the reflectance field, were introduced as continuous functions. Since we want to sample these functions and use the measurements for rendering, discrete representations are needed. Therefore, we discretize the domain of the incoming light field by defining a set of basis illuminations $\mathcal{L} = \{\mathbf{l}_i\}_{i \in I}$ indexed by an index set I. Furthermore, we denote by $\mathbf{r}_i$ the discretized outgoing light-field (parameterized by an index set K) which is the scene's response to the incoming basis illumination $\mathbf{l}_i$. Then, due to the linearity of light transport the outgoing light field $\mathbf{l}_o$ for an arbitrary, linear combination $\mathbf{l} = \sum_i l_i \mathbf{l}_i$ of basis illuminations is given as

$$\mathbf{l}_o = \sum_{i \in I} l_i \mathbf{r}_i \tag{2.13}$$

which can be written in matrix form:

$$\mathbf{l}_o = \mathbf{R}^{(K \times I)} \mathbf{l} \tag{2.14}$$

where $\mathbf{R}^{(K \times I)} = [\mathbf{r}_1 \mathbf{r}_2 \dots \mathbf{r}_{|I|}]$ is the $|K| \times |I|$- matrix of the concatenated basis light field response vectors which is usually called the *light transport matrix*.

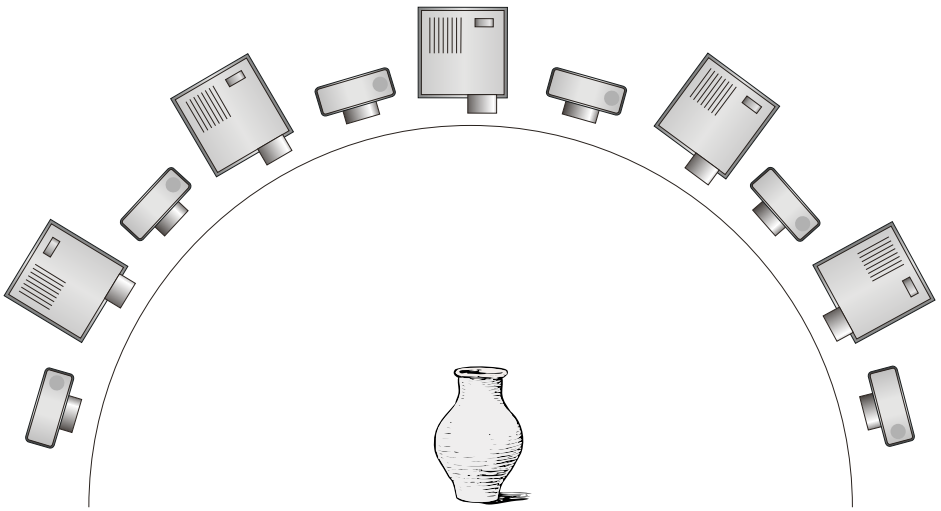


Figure 2.4: Schematic acquisition of the 8D transport matrix. An array of cameras samples the outgoing light field. Arbitrary incident light fields are generated by an array of projectors.

Equation (2.14) can be interpreted as the discrete version of the relighting equation (2.11).

In this respect, the sampled BTF can be defined as a reduced version of the full transport matrix by assuming only infinitely distant lighting:

$$\mathbf{l}_o = \mathbf{B}^{(K \times I^d)} \mathbf{l}^d \tag{2.15}$$

Each column $\mathbf{b}_j$ of $\mathbf{B}^{(K \times I^d)}$ represents the outgoing light field of the captured scene in response to the directional light source $\mathbf{l}_i^d$ indexed by $i \in I^d$. The rows $\mathbf{b}^j$ are so-called 2D *reflectance functions* [DHT+00] - they represent the response of a specific sample from the outgoing light field to the varying basis lights.

2.6.2 Capturing the Light Transport Matrix

Measuring $\mathbf{R}^{(K \times I)}$ for a specific scene corresponds to lighting the scene with each basis light $\mathbf{l}_i$ and measuring the corresponding outgoing light field (using an appropriate detector). A hypothetical device for measuring the full LTM is depicted in Figure 2.4. It consists of a set of projectors which are placed along a hemispherical bounding surface and are able to cast light rays from discrete positions into a discrete set of directions by switching individual pixels. The outgoing light-field is captured by perspective cameras placed along the same bounding hemisphere.

While it seems possible to build such a setup physically, a quick calculation suggests that a dense sampling of $\mathbf{R}^{(K \times I)}$ seems out of reach. Assuming a reasonable sampling of 512×512 projector/camera pixels and 32×32 positions on the hemisphere results in a transport matrix with about $(2.68 \cdot 10^8)^2$ entries!

Therefore, most researchers concentrated on measuring slices of the full 8D transport matrix. Capturing only the outgoing light field for one fixed illumination ($|I| = 1$) is the most common case (e.g. [LH96, GGSC96, MRP98, WAA+00]). In their seminal paper Levoy & Hanrahan captured between 256 and 2048 images of 256^2 pixel resolution which took up to 4 hours of acquisition time. Another well-known 4D slice is captured by the first Light Stage from Debevec et al. [DHT+00]. They assumed directional lighting and a fixed viewpoint. Using a rotatable light arm and a video camera they where able to capture about 64×32 different light directions in one minute.

The first work which dealt with near-field illumination was published by Masselus et al. [MPDW03]. They used projectors to generate $224 \times 256 = 57344$ different basis incident light fields which took more than 100 hours to capture for a single view.

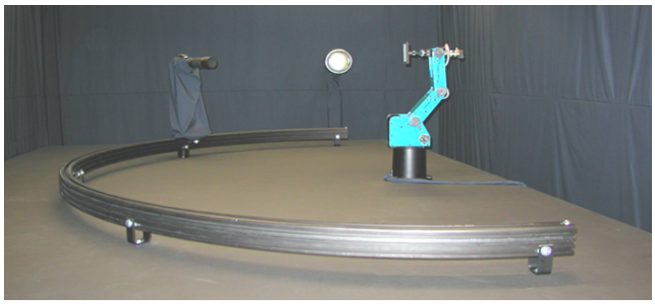
By the time of writing this thesis the only work which tried to tackle the full problem is by Garg et al. [GTLL06]. They used only a vary sparse directional sampling (3×3 directions which cover only a small sector of the hemisphere and 130×200 pixel spatial resolution) and exploited a hierarchical representation of the transport matrix to reduce the actually required number of samples by two orders of magnitude. While their results are already impressive just extending the setup to cover the whole hemisphere of directions would result in measurement times in the range of weeks.

2.6.3 BTF Acquisition

BTF-Measurement fits into the general light transport acquisition framework from Figure 2.4 with the difference that the light projectors are replaced by directional light sources. Thereby, the acquisition complexity is reduced by two dimensions. Furthermore, the samples are usually almost planar which simplifies the required calibration and registration procedures significantly.

The BTF measurements used in this thesis have been captured by two different measurement devices built at the University of Bonn.

The first setup, published in [SSK03], is based on a classical gonioreflectometer-like configuration which means light source and detector can be placed freely on the hemisphere above the sample. It is realized by placing the sample on a robot-arm which moves and rotates the sample in space. The light source is an HMI bulb placed about 2.5 m away from the sample in order to achieve approximately directional lighting. The camera (a Kodak DCS Pro 14M) is mounted on



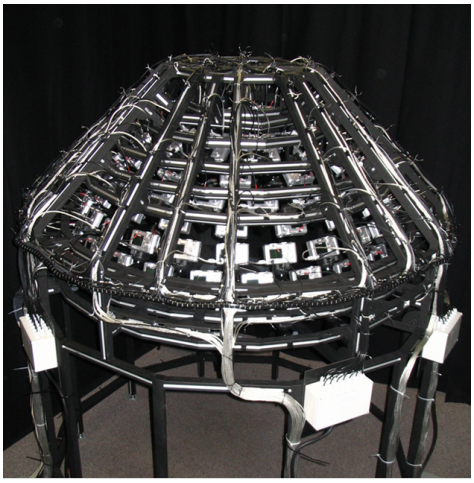
θ [°]	$\Delta\phi$ [°]	#images
0	–	1
15	60	6
30	30	12
45	20	18
60	18	20
75	15	24

Figure 2.5: The gonioreflectometer-like setup at University of Bonn(left). The sampling of the viewing and illumination angles of the measurements used in this thesis sums up to 81 angles for each hemisphere (right).

half-circle rail at about the same distance from the sample (see Figure 2.5 left). While the setup in principle allows arbitrary samplings of the lighting and viewing directions all measurements used in this thesis are using the sampling depicted in Figure 2.5 which results in 81 directions per hemisphere and thus in 6561 images for a complete measurement.

The second setup was published in [RMS+08]. It aims at speeding up the acquisition process by using a set of cheap consumer cameras in parallel. In the realized version 151 cameras with a resolution of 3.2 megapixels were mounted on a hemispherical aluminum gantry. Figure 2.6 shows a photograph of the setup and gives the positions of the cameras. The radius of the hemisphere is about 80 cm and the built-in flash lights are employed as light sources. Due to the relatively short distance from the sample the light sources cannot be assumed to be directional which requires a correction step after measurement.

Discussing further details and practical issues of these BTF measurement devices lies not within the scope of thesis. We refer the interested reader to the state-of-the-art report about BTF acquisition, synthesis and rendering [MMS+05].



θ [°]	$\Delta\phi$ [°]	#images
0	–	1
11	60	6
23.5	30	12
30	30	12
37.5	30	12
45	20	18
52.5	20	18
60	15	24
67.5	15	24
75	15	24

Figure 2.6: The parallel BTF acquisition device built at University of Bonn (left). The device consists of 151 cameras which are arranged on the hemisphere as shown in the table on the right. Since the built-in flash lights are used as light sources a maximum of 151 samples can be measured for each hemisphere.

Part I

Compression

CHAPTER 3

BACKGROUND

The storage requirements of uncompressed measured appearance data easily exceed several gigabytes which makes compression a basic necessity of any image-based rendering system. Such compression techniques should not only allow *high compression ratios* while maintaining most of the visual fidelity of the original data but also a *fast and random access decompression* step for efficient rendering. Unfortunately, the well-known techniques from the field of image and video compression do not fulfill these demands.

For example with lossless compression techniques based on entropy coding [Sha48] compression ratios of about 3:1 can be achieved (cf. Figure 4.1) which is obviously not sufficient. Applying sophisticated lossy image compression algorithms like the ones used within the JPEG or JPEG2000 standards to a single or a block of images sounds attractive but is in fact impractical, since the decompression stage is too expensive for rendering applications and the achievable compression ratios of about 1:20 are still not high enough considering raw data sizes of several GBs.

In order to achieve the above goals so-called *domain specific* algorithms are required which exploit the special structure and properties of image-based data and in particular BTF data. The huge amount of literature in this area can be roughly categorized into three different groups:

Parametric modeling - adapting ideas from BRDF- and/or texture modeling to spatially varying reflectance data

Basis projection - using suitable basis functions like multi-dimensional Wavelets or Spherical Harmonics

Statistical data analysis - computation of customized, *data-driven* basis functions using techniques like Principal Component Analysis (PCA)

Existing techniques based on parametric modeling and basis projection using predefined basis function will be reviewed in Chapter 6. In this and the following

chapter we will focus on the latter group which is statistical data analysis. Within this group we will discuss those techniques which basically boil down to the application of matrix factorization algorithms to the light transport matrices (2.14) and (2.15) which explains the title of Chapter 4.

A main advantage of these techniques compared to parametric modeling and fixed basis functions is their greater *generality* and *effectiveness* thanks to the data-driven basis functions.

Generality in this context means that the basis is adapted to the particular dataset and that in contrast to parametric models, which are usually designed to model a specific class of materials (e.g. uniform metals or plastic), almost no restricting assumptions about the material classes that can be represented are made. In fact, it is only assumed that the data matrix has a *simple covariance structure* which means a covariance matrix of low rank.

Thereby, these methods are not only general but also effective since in contrast to using a-priori fixed basis functions like Spherical Harmonics the data-driven basis adapts to the data which usually enables better compression ratios and lower runtime reconstruction costs. Since in contrast to a fixed basis the data-driven basis has to be stored together with the coefficients there is an overhead involved but this overhead is usually more than compensated by the less coefficients required.

However, it is also not clear a-priori that the light transport matrices in general have a simple covariance structure and that few custom basis functions suffice for an efficient approximation. In fact, simple empirical tests show that for complicated materials the Eigenvalues of the covariance matrix decrease slowly which suggests that spatially varying light transport in general is a high-dimensional phenomenon. Furthermore, the computation of the data-driven basis functions can pose a serious computational challenge because of the huge size of the data matrix. As a result several variants of the basic matrix factorization scheme have been introduced over the years, among them combinations of factorization and clustering (e.g. [MMK03]) and tensor factorization techniques (e.g. [WWS+05]) which essentially correspond to the application of matrix factorization to different subsets and arrangements of the data.

In this part of the thesis we will compare various of these different factorization techniques in terms of compression ratio, reconstruction quality and run-time decompression cost. Additionally, we also analyze the performance of two-way clustering of matrix elements which has not yet been applied for BTF compression. Furthermore, in Chapter 5 we introduce *Data-Driven Local Coordinate Systems* – a technique which improves the correlation between rows and columns of the data matrix by estimating local frames per surface point. This improves correlation and results in a decreased rank of the covariance matrix which then results in an improved compression ratio of matrix factorization techniques.

In the remainder of this background chapter we provide the theoretical foun-

dations of matrix and tensor factorization techniques independently of our specific application area (compression of image-based measurements).

3.1 Matrix Factorization

Matrix *factorization* or *decomposition* is a quite general concept from linear algebra. In its general form a factorization of the matrix $\mathbf{A}$ is a product of the form

$$\mathbf{A} = \prod_{i}^{K} \mathbf{F}_i.$$

The key role of matrix factorizations in numerical computations stems from the fact that factors of special structure can help tremendously in solving certain matrix problems. A famous example is the *LU-factorization* for invertible matrices where $\mathbf{A} = \mathbf{LU}$ and $\mathbf{L}$ is lower-triangular and $\mathbf{U}$ is upper-triangular. The LU-factorization can be used to efficiently solve linear equations $\mathbf{Ax} = \mathbf{LUx} = \mathbf{b}$ without gaussian-elimination simply by forward and backward substitution. If the linear system has to be solved several times for different $\mathbf{b}$ the cost for the factorization itself quickly amortizes.

The importance of matrix factorizations for data compression comes in the form of the *Singular Value Decomposition* which will be discussed in the following subsection where we will also give the relation between SVD and PCA which in fact can be used synonymously in case of mean-centered data matrices. As mentioned in the introduction the light transport matrix generally has high rank which implies high storage requirements and reconstruction cost. Furthermore, the factorization of huge matrices can pose a huge computational challenge. This motivates *blockwise*- and *clustered* factorization techniques which aim at finding local low-rank approximations. Another trend is *tensor factorization* which generalizes matrix factorization to higher dimensional arrangements of data. Thereby, redundancies along this additional dimensions could be exploited, eventually resulting in a more compact factorization.

3.1.1 Singular Value Decomposition

Because of its fundamental importance we will give the SVD-theorem here. We assume that the reader is familiar with the basic concepts from linear algebra like *rank*, *eigen-values* and *eigen-vectors* as well as *unitary* and *diagonal* matrices.

Theorem 3.1. (SVD)
If $\mathbf{A} \in \mathbb{R}^{m \times n}$ has rank k, then it may be written in the form

$$\mathbf{A} = \mathbf{USV}^T$$

where $\mathbf{U} \in \mathbb{R}^{m\times m}$ *and* $\mathbf{V} \in \mathbb{R}^{n\times n}$ *are unitary matrices. The matrix* $\mathbf{S}$ *is a* $m \times n$*-diagonal matrix with* $s_{ii} \geq s_{i+1,i+1} > 0$ *for* $1 \leq i < k$ *and* $s_{ii} = 0$ *for* $k+1 \leq i \leq \min(m,n)$*. The numbers* s_{ii} *are the nonnegative square roots of the eigenvalues of* $\mathbf{A}\mathbf{A}^T$*. The columns* $\mathbf{u}_j$ *of* $\mathbf{U}$ *are the eigenvectors of* $\mathbf{A}\mathbf{A}^T$ *and the columns* $\mathbf{v}_j$ *of* $\mathbf{V}$ *are the eigenvectors of* $\mathbf{A}^T\mathbf{A}$*.*

For a proof of Theorem 3.1 we refer the interested reader to [HJ86, p.415]. The crucial property of the SVD for data-compression is now given by the Eckart-Young Theorem:

Theorem 3.2. (Eckart-Young)
Let $\mathbf{U}\mathbf{S}\mathbf{V}^T$ *be the SVD of the rank-*k *matrix* $\mathbf{A} \in \mathbb{R}^{m\times n}$ *as given in Theorem 3.1. Furthermore, we define for* $r \leq k$ *the* rank-r approximation $\mathbf{A}_r$ *of* $\mathbf{A}$ *as follows:*

$$\mathbf{A}_r := \sum_{j}^{r} \mathbf{u}_j s_{jj} \mathbf{v}_j^T$$

Then $\mathbf{A}_r$ *is the best rank-*r *approximation of* $\mathbf{A}$ *in the least-squares sense:*

$$\mathbf{A}_r = \underset{rank(\mathbf{B})=r}{\arg\min} |\mathbf{A}-\mathbf{B}|_F \text{ with } |\mathbf{C}|_F = \sqrt{\sum_{i}^{m}\sum_{j}^{n} c_{ij}^2}$$

Proof. The key is that the Frobenius-norm $|\bullet|_F$ is unitarily invariant: $|\mathbf{U}\mathbf{A}\mathbf{V}^{\mathbf{T}}|_F = |\mathbf{A}|_F$ if $\mathbf{U}$ and $\mathbf{V}$ are unitary. After Theorem 3.1 the SVD of $\mathbf{A} = \mathbf{U}\mathbf{S}\mathbf{V}^T$ exists and we have:

$$|\mathbf{A}-\mathbf{B}|_F^2 \;=\; |\mathbf{S}-\mathbf{U}^T\mathbf{B}\mathbf{V}|_F^2$$

Since $\mathbf{S}$ is a diagonal matrix, for any minimizer of $|\mathbf{A}-\mathbf{B}|_F^2$ the matrix $\mathbf{D} = \mathbf{U}^T\mathbf{B}\mathbf{V}$ must be a diagonal matrix, too. Therefore, $\mathbf{U}\mathbf{D}\mathbf{V}^T = \mathbf{B}$ is the singular value decomposition of $\mathbf{B}$. Since $\mathbf{B}$ ought to be of rank r, $\mathbf{D}$ has only r non-zero entries. It follows:

$$\min_{rank(\mathbf{B})=r} |\mathbf{A}-\mathbf{B}|_F^2 = |\mathbf{S}-\mathbf{D}|_F^2 = \min_{d_{ii}, 0<i\leq r} \left(\sum_{i}^{r} (s_{ii}-d_{ii})^2 + \sum_{i=r+1}^{n} s_{ii}^2 \right)$$

which implies $d_{ii} = s_{ii}$ for $0 < i \leq r$. □

An obvious application of Theorem 3.2 is data compression since the original storage requirements of $\mathcal{O}(mn)$ can be reduced to $\mathcal{O}((m+n)r)$ which results in significant savings if $r \ll m, n$ since the compression ratio is $(\frac{r}{n} + \frac{r}{m})$.

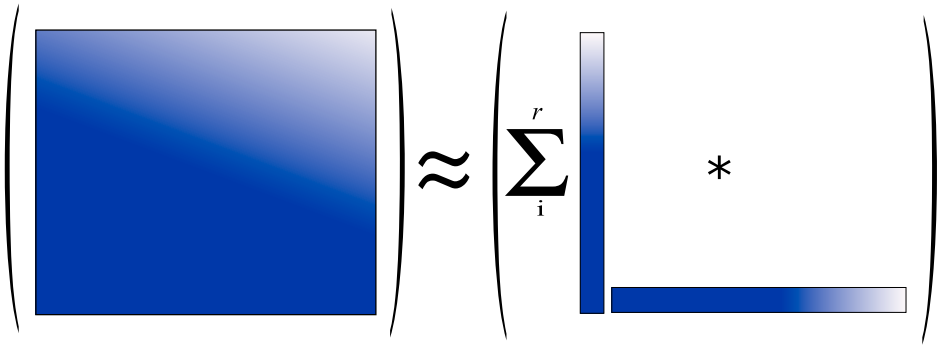


Figure 3.1: Factorizing a matrix into a sum of outer-products (rank-1 matrices).

The price to be paid is the reconstruction error $\sum_{i=r+1}^{n} s_{ii}^2$ which can be interpreted as the variance not captured in the rank-r approximation since as given above the s_{ii} are the square roots of the eigenvalues of the covariance matrix $\mathbf{A}\mathbf{A}^T$.

Generally, we can state that a matrix $\mathbf{A}$ can be well reconstructed by a short sum of outer products (rank-1 matrices) as illustrated in Figure 3.1 if the eigenvalues of $\mathbf{A}\mathbf{A}^T$ decay fast, i.e. the correlation between the rows/columns of $\mathbf{A}$ can be considered sufficiently high. Please note, that computing the SVD of $\tilde{\mathbf{A}}$ where the column mean $\bar{\mathbf{a}} = \frac{1}{n}\sum_i^n \mathbf{a}_i$ is subtracted from each column is equivalent to performing a principal component analysis (PCA, see e.g. [Jol86]) on the column vectors of $\mathbf{A}$. In this case $\mathbf{U}$ is the matrix of eigenvectors and $\mathbf{S}\mathbf{V}^T$ is the matrix of PCA-weights. In the following we will always assume mean-centered data matrices.

The computation of the SVD is usually performed by computing the symmetric row or column covariance matrix $\mathbf{A}\mathbf{A}^T \in \mathbb{R}^{m\times m}$ or $\mathbf{A}^T\mathbf{A} \in \mathbb{R}^{n\times n}$ depending on which is smaller. Then an eigen-decomposition algorithm for symmetric matrices like the Jacobi method can be used to compute $\mathbf{U}$ or $\mathbf{V}$ respectively and $\mathbf{S}^2$. If we assume the case $m < n$ then $\mathbf{V}^T$ can be computed by $\mathbf{V}^T = \mathbf{S}^\dagger\mathbf{U}^T\mathbf{A}$ where $\mathbf{S}^\dagger$ denotes the pseudo-inverse of $\mathbf{S}$.

The complexity for computing the covariance is $\mathcal{O}(m^2n)$, the diagonalization requires $\mathcal{O}(m^3)$ and the computation of $\mathbf{V}$ $\mathcal{O}(rnm)$ operations if only the first r columns of $\mathbf{V}$ are computed. Since we assumed $m < n$ the overall factorization complexity is bounded by the cost of the covariance matrix computation. Therefore, numerous approximation algorithms have been developed which try to avoid this costly step. Examples are *Online-SVD* [Bra03] or *Expectation-Maximization-PCA* [Row98] which compute approximate representations of the orthogonalized sub-spaces iteratively. These algorithms have usually a complexity of $\mathcal{O}(rnm)$ which leads to huge computational savings if r is small which is usually the case

in compression applications. In our practical experiments we used EM-PCA because of its ease of implementation and numerical stability. We also tried Online-SVD which behaves similarly in terms of storage requirements and computational complexity but it has the disadvantage of requiring a threshold parameter which decides if the current subspace has to be extended and which turned out to be difficult to choose appropriately in practice.

3.1.2 Block-Wise Factorization

Applying SVD to the complete matrix $\mathbf{A}$ can be interpreted as a *global factorization*. A common approach to achieve lower rank factorizations of large matrices is to perform a *local* or *block-wise factorization* by partitioning the matrix into sub-blocks which are factorized independently (e.g. [MPN+02, CBCG02, MMK03, NBB04, NNJ05, GTLL06]). Usually, it can be assumed that these sub-blocks have a lower rank than the whole matrix. In fact, it has been shown in [MSRB07] that there is linear relationship between the dimensionality of the light transport matrix and the spatial patch size. The various block-wise matrix factorization techniques used in compression of image-based data differ mainly in the way they select and organize matrix-blocks. We will present these techniques in a common notational framework for block matrices following [Hac99].

It is convenient to define what we mean with a partitioning:

Definition 3.1. (Partitioning)
Let I be a finite index set (e.g. $I = \{1, \ldots, n\}$) with $|I| = n$. Then $P_1 = \{I_j : 1 \leq j \leq p_n\} \subset \mathcal{P}(I)\backslash\{\emptyset\}$ is a partitioning *of I if*

$$\begin{aligned} I_i \cap I_j &= \emptyset \quad \textit{for} \quad i \neq j \\ I &= \bigcup_{j=1}^{p_n} I_j. \end{aligned}$$

The corresponding j-th vector block of a vector $\mathbf{a} \in \mathbb{R}^n$ is $(a_i)_{i \in I_j}$

It is important to note that according to this definition the index blocks I_j do not need to consist of consecutive elements. They just represent an arbitrary grouping of index elements. In order to find the block I_j containing an index element i we have to store a block index per element in the general case. Block partitioning generalizes this concept to 2-dimensional index sets:

Definition 3.2. (Block Partitioning)
Let I, J be finite index sets with $|I| = m$ and $|J| = n$ (e.g. $I = \{1, \ldots, m\}$ and

$J = \{1, \ldots, n\}$). *Then* $P_2 = \{b_j : 1 \leq j \leq p\} \subset \mathcal{P}(I \times J) \backslash \{\emptyset\}$ *is a* block partitioning *of* $I \times J$ *if*

$$\begin{aligned} b_i &= \tilde{I} \times \tilde{J} \quad \textit{with } \tilde{I} \subset I \textit{ and } \tilde{J} \subset I \\ b_i \cap b_j &= \emptyset \quad \textit{for} \quad i \neq j \\ I \times J &= \bigcup_{j=1}^{p} b_j. \end{aligned}$$

The matrix block of a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ *corresponding to* $b_k \in P_2$ *is given as* $(a_{ij})_{(i,j) \in b_k}$

This definition allows to partition the index set $I \times J$ into arbitrary, rectangular blocks - they just have to cover the whole index set and are not allowed to overlap. Again, in the general case we have to store a block index per index element in order to find its containing block in constant time. Therefore, in practice more restricted block schemes are used which minimize the overhead for storing and accessing the block structure. We will now shortly introduce the most common schemes in order of their complexity which are *regular blocks* and *tensor-product blocks*.

Definition 3.3. (Regular Block Partitioning)
Let the finite index sets I, J *with* $|I| = m$ *and* $|J| = n$ *and corresponding partitionings* P_I *and* P_J *be given according to Definition 3.1. Then the block partitioning* $P_r = \{I_i \times J_j : I_i \in P_I, J_j \in P_J\}$ *is a* regular block partitioning *if there exist two integers* m_b *and* n_b *such that* $p_m = m/m_b$ *and* $p_n = n/n_b$ *are integers and*

$$\begin{aligned} |I_i| &= m_b \quad \textit{for } I_i \in P_I \\ |J_j| &= n_b \quad \textit{for } J_j \in P_J \end{aligned}$$

If $m \bmod m_b \neq 0$ *or* $n \bmod n_b \neq 0$ *then there exists also one part* $I_{i_0} \in P_I$ *with* $|I_{i_0}| = m \bmod m_b$ *or one part* $J_{j_0} \in P_J$ *with* $|J_{j_0}| = n \bmod n_b$ *respectively.*

Usually, it is assumed that the index sets are ordered and the partitions P_I and P_J contain consecutive elements ($\max(I_i) < \min(I_{i+1})$). In this case the block index for an index element (i, j) can be computed simply as $(\lfloor i/m_b \rfloor, \lfloor j/n_b \rfloor)$.

The number of blocks is given by $\lceil p_m \rceil \times \lceil p_n \rceil$. If SVD is now applied to each block as illustrated in Figure 3.2 and a single rank r_b is chosen for all blocks the storage requirements are given by $\mathcal{O}(p_m p_n (m_b + n_b) r_b)$.

Regular blocks with consecutive elements can be accessed in constant time which leads to $\mathcal{O}(r_b)$ reconstruction cost per matrix element and the compression ratio is given by $r_b(\frac{p_m}{m} + \frac{p_n}{n})$.

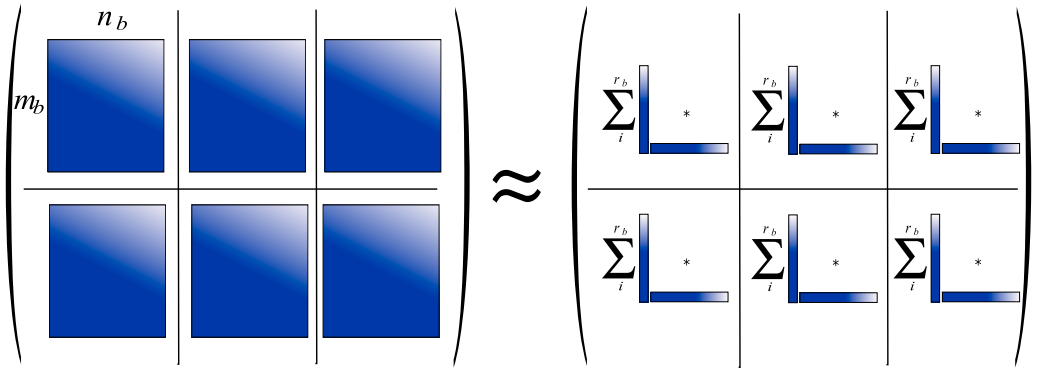


Figure 3.2: Principle of factorizing regular blocks: Each regular block is independently approximated by a low-rank factorization.

It can easily be seen that $r_b \cdot p_m$ and $r_b \cdot p_n$ have to be in the order of r if the compression ratio should be comparable to standard factorization. If reconstruction cost is of more prominent importance a slightly worse compression ratio can be accepted if r_b is only low enough such that a certain rendering performance is achieved. For example, in the case of $p_m = p_n = 10$, r_b must be in the range of $0.1r$ to reach the same compression ratio. If the resulting reconstruction error is too high we probably might need to double the number of components and hence double the memory requirements but even then the reconstruction cost will be approximately five times lower compared to standard SVD.

The computational complexity of block factorization is similar to standard factorization if an approximation algorithm like Roweis' EM-algorithm is used (cf. previous section).

While a significant rank reduction can be achieved already by simple block-wise factorization it has to be noted that the block size and position are fixed beforehand. One problem of this approach is well-known from block-based image- or video compression techniques like JPEG or MPEG-2: the regular borders between the independently approximated blocks are easily spotted as *block artifacts* in the reconstructed images. Another problem is that a possibly great deal of coherence between elements from different blocks is not exploited.

By relaxing the restriction of a fixed block size tensor-product partitionings allow better adaption of block shapes and sizes to the data:

Definition 3.4. (Tensor-Product Partitioning)
Let the finite index sets I, J with $|I| = m$ and $|J| = n$ and corresponding partitionings P_I and P_J be given according to Definition 3.1. Then the block partitioning $P_r = \{I_i \times J_j : I_i \in P_I, J_j \in P_J\}$ is a tensor-product partitioning.

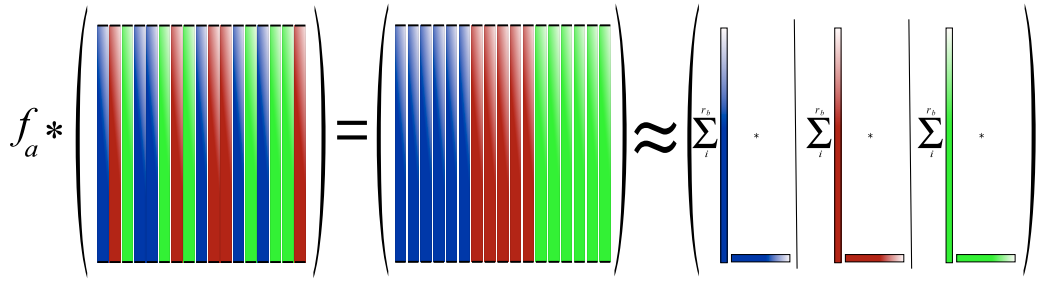


Figure 3.3: Principle of adaptive factorization using only column partitioning: a column partition P_J is computed (here f_a symbolizes a matrix which permutes the columns of $\mathbf{A}$ to get the matrix of the concatenated block matrices $\mathbf{A}_{J_i}$). Then the block matrices $\mathbf{A}_{J_i}$ are factorized independently.

Of course, the regular block partitioning is a special case of the tensor-product partitioning with consecutive blocks of equal size. General tensor-product blocks are of varying size and their partial rows and columns can be scattered across the whole matrix. Therefore, finding the block of a given matrix element generally requires to store a row- and column block index for the elements of the partitions P_I and P_J which requires $\mathcal{O}(m+n)$ storage space. Then the tensor-product block index can be looked-up in constant time and the reconstruction cost remains $\mathcal{O}(r_b)$ if again a fixed rank r_b per sub matrix is used.

Given a number of row partitions $p_m = |P_I|$ and a number of column partitions $p_n = |P_J|$ the storage requirements for tensor-product block factorization are similar to the regular block case and the compression ratio is approximately the same (up to the block indices). The computational complexity for the factorization alone remains $\mathcal{O}(mnr_b)$ if an iterative method is used.

Adapting Blocks to Matrix Content

The advantage of tensor-product blocks compared to regular blocks is that they can be adapted to the matrix content. We will call these kind of block schemes *adaptive*. Ideally, we want to find partitions P_I and P_J which are optimal in terms of reconstruction cost, reconstruction error and storage requirements. Unfortunately, the number of possible partitions of an index set I of size n is given by the *Bell*-numbers B_n [Com74] which grow super-exponentially (E.g. a set of 20 elements has already more than 51 trillion possible partitions). Hence, we cannot hope to find such partitions by exhaustive search.

In practice, some feasible error criterion is defined and then clustering algorithms are used to minimize this measure. If we allow to partition only the col-

umn space ($|P_I| = 1$) as illustrated in Figure 3.3 we have the classical problem of clustering a set of n m-dimensional data vectors. Upon the numerous existing algorithms for solving this problem the *k-means* algorithm [Mac67] is probably the most popular one (at least for a quadratic error criterion).

The more general problem of simultaneously clustering rows and columns is known in the literature as *two-way*- or *co*-clustering and it has been studied mainly in the field of bioinformatics, in particular gene expression analysis (e.g. [GLD00] [DMM03]).

To this end we will only detail out the classical *one-way* clustering approach on the basis of the k-means algorithm. It computes a local minimum of the following cost function:

$$E_{P_J} = \sum_{i=1}^{p_n} \sum_{j \in J_i} (\mathbf{a}_j - \bar{\mathbf{a}}_{J_i})^2 \tag{3.1}$$

where $\bar{\mathbf{a}}_{J_i} = \sum_{j \in J_i} \mathbf{a}_j / |J_i|$ is the mean of the column block J_i. Typically, a local minimum of (3.1) is computed using Lloyd's algorithm [GG91]. Then matrix blocks $\mathbf{A}_{J_i}$ can be collected and SVD can be applied to each of them:

$$\mathbf{A}_{J_i} = (\mathbf{a}_j)_{j \in J_i} \approx \sum_{j=1}^{r_b} \mathbf{u}_{J_i,j} \mathbf{w}_{J_i,j}^T \tag{3.2}$$

The standard k-means clustering approach can be improved by using the reconstruction error in the error function as proposed by Kambhatla and Leen [KL97]:

$$E_{P_J} = \sum_{i=1}^{p_n} \sum_{j \in J_i} \left(\mathbf{a}_j - \bar{\mathbf{a}}_{J_i} - \sum_{k=1}^{r_b} \langle \mathbf{a}_j - \bar{\mathbf{a}}_{J_i}, \mathbf{u}_{J_i,k} \rangle \mathbf{u}_{J_i,k} \right)^2 \tag{3.3}$$

In this case the computation of the centroid in the classical Lloyd iteration is replaced by the factorization of the block matrix $\mathbf{A}_{J_i}$. In order to initialize the per-block subspaces a few standard Lloyd iterations are performed.

The time complexity of a single Lloyd iteration can easily be estimated: Every data point has to be checked against the p_n cluster centers or sub-spaces respectively which sums up to $\mathcal{O}(nmp_n(r_b + 1))$ operations ($r_b = 0$ for basic k-means). The per-cluster subspaces have to be recomputed in each iteration which has complexity $\mathcal{O}(nm(r_b + 1))$ with r_b as above. Despite the fact that in the general case a fast convergence cannot be guaranteed (finding the exact solution of Equation 3.1 is NP-hard) the number of iterations actually needed is usually quite low in practice.

3.1.3 Two-pass Factorization

While it is possible to reduce the coherence between independently factorized blocks by clustering there will usually remain some sort of redundancy between

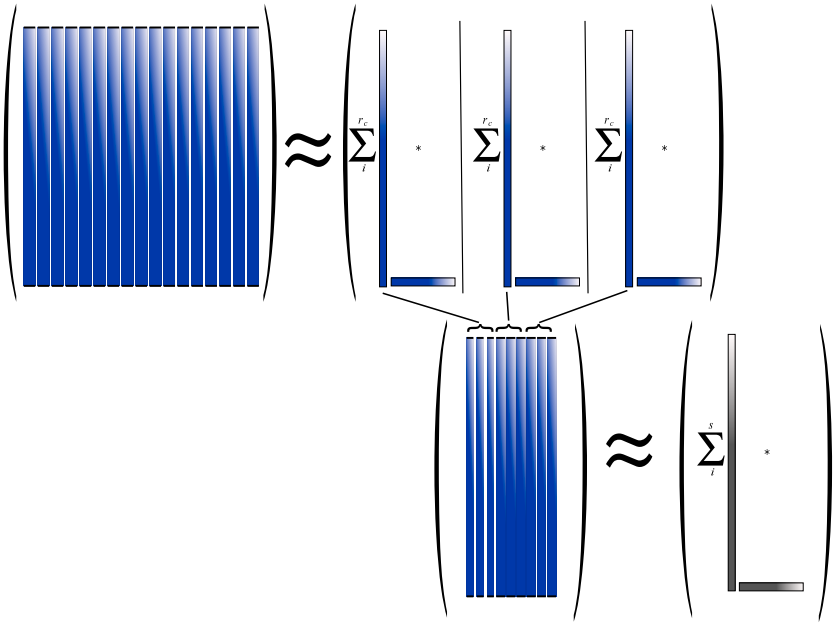


Figure 3.4: Two-pass factorization simply takes all basis vectors (in this case the column-space basis) from each block and applies an additional factorization step.

the subspaces. In order to exploit this redundancy Nayar et al. [NBB04] proposed to perform a second factorization pass on the basis vectors of the already computed subspaces as illustrated in Figure 3.4. The motivation is that if a regular block partitioning is used redundancy between blocks will be rather likely.

We will formulate the method based on a simple block scheme with m_b rows per-block and $p_m = \frac{m}{m_b}$ blocks which is in the spirit of the original paper. We omit the straight-forward generalization to a general block scheme here for brevity.

Factorizing each block $\mathbf{A}_b \in \mathbb{R}^{m_b \times n}$ yields

$$\mathbf{A}_b \approx \tilde{\mathbf{U}}_b \tilde{\mathbf{W}}_b^T$$

with $\tilde{\mathbf{U}}_b \in \mathbb{R}^{m_b \times r_b}$ being the reduced orthogonal block column basis and $\tilde{\mathbf{W}}_b = \tilde{\mathbf{V}}_b \tilde{\mathbf{S}}_b \in \mathbb{R}^{n \times r_b}$ being the weight matrix describing how each column will be reconstructed (reduced orthogonal block row basis times singular values).

Concerning storage requirements it can be immediately observed that this type of factorization needs to store a full row basis for every block leading to $\mathcal{O}((m + np_m)r_b)$ storage requirements. If we assume that there is redundancy between rows over the whole matrix there will most likely be also some inter row-base redundancy which means that the matrix $\mathbf{X} \in \mathbb{R}^{n \times r_b \cdot p_m}$, which is the

concatenation of all per block weight matrices, has a low-rank representation:

$$\mathbf{X} \approx \tilde{\mathbf{Y}}\tilde{\mathbf{Z}}^{\mathbf{T}}$$

with $\tilde{\mathbf{Y}} \in \mathbb{R}^{n \times s}$ and $\tilde{\mathbf{Z}} \in \mathbb{R}^{r_b \cdot p_m \times s}$. This way the total storage requirements are reduced to $\mathcal{O}(mr_b + (n + p_m r_b)s)$ which pays off if n and $p_m r_b$ are large compared to s.

The price to pay is the increased reconstruction cost since the weight matrix $\tilde{\mathbf{W}}_b = \tilde{\mathbf{Y}}\tilde{\mathbf{Z}}_b^T$ has to be reconstructed during run-time which leads to costs of $\mathcal{O}(r_b s)$ per single matrix element.

3.1.4 Tensor Factorization

All factorization techniques mentioned so far operate on a 2D-matrix of measurements. Since image-based measurements usually have higher dimensionality the data elements have to be squeezed somehow into this 2D corset. The definition of the discrete BTF transport matrix in Equation 2.15, for instance, showed one way how this can be done for the 6-dimensional BTF.

In the context of data compression this approach has the consequence that some *intra*-row or -column correlations are not taken into account. Consider for example the typical way of arranging an image sequence into a matrix which is to unroll the rows and columns of each single measurement image into a vector and concatenating these vectors into a matrix. If SVD is applied to this matrix, intra-image correlations, like the correlations between the rows and columns of the images, are not fully considered. As demonstrated above, the problem can be alleviated using block-wise factorization techniques (which try to minimize the correlations between blocks) but also these techniques rely on the original 2D arrangement of the high-dimensional data.

The in a sense more *natural* solution here are tensors. The idea is to arrange the data in multi-dimensional grids which naturally generalize the familiar matrix concept. For instance a three-dimensional tensor can be easily imagined by humans as a 3D-data cube. More formally, an N-th order tensor is denoted in calligraphic letter as $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ with $I_n \in \mathbb{N}^+$ denoting the size of each dimension. Elements of $\mathcal{A}$ are accessed using N-dimensional index tuples and denoted as $a_{i_1 \ldots i_n \ldots i_N}$ with $1 \leq i_n \leq I_n$. The question now immediately arises if there exists a low-rank factorization of general tensors which fulfills optimality criteria similar to the SVD for matrices (which can be interpreted as 2nd order tensors). The answer is that an equivalent to the Eckart-Young theorem for tensors has not been found yet, but fortunately there are algorithms which at least find a local optimum. Among the most popular of these algorithms in the field of image-based rendering and compression is the *higher-order* SVD (HOSVD) based on N-mode SVD and an alternating least-squares (ALS) method [LMV00].

We will introduce only the basic definitions from multi-linear algebra needed for the formulation of the algorithm and refer the interested reader to [LMV00] and [WWS+05] for any further details. The concepts we need to know are the mode-n vectors and matrices of a tensor as well as the mode-n product of a tensor and a matrix.

Definition 3.5. (Mode-n vector, mode-n matrix)
The mode-n vector $\mathbf{v}$ *of a tensor* $\mathcal{A}$ *is an* I_n*-dimensional vector consisting of the elements* $(a_{i_1 \dots i_n \dots i_N})_{1 \leq i_n \leq I_n}$ *of* $\mathcal{A}$ *where only the index* i_n *varies while all other indices stay fixed. The mode-n matrix* $\mathbf{A}_{(n)} \in \mathbb{R}^{I_n \times (I_1 \cdot I_2 \cdot \dots \cdot I_{n-1} \cdot I_{i+1} \cdot \dots \cdot I_N)}$ *consists of all mode-n vectors and can be interpreted as the* flattening *of the tensor along its n-th dimension.*

Definition 3.6. (Mode-n product)
The mode-n product between a tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_N}$ *and a matrix* $\mathbf{U} \in \mathbb{R}^{J_n \times I_n}$ *is a tensor* $\mathcal{B} = (\mathcal{A} \times_n \mathbf{U}) \in \mathbb{R}^{I_1 \times \dots \times I_{n-1} \times J_n \times I_{n+1} \dots \times I_N}$*. Its entries are given by* $b_{i_1 \dots i_{n-1} j_n i_{n+1} \dots i_N} = \sum_{i_n} a_{i_1 \dots i_n \dots i_N} u_{j_n i_n}$ *and can be expressed in terms of flattened mode-n matrices as* $\mathbf{B}_{(n)} = \mathbf{U}\mathbf{A}_{(n)}$.

We can now formulate the classical SVD within the tensor framework. The matrix $\mathbf{A} \in \mathbb{R}^{I_1 \times I_2}$ is a second order tensor with two modes. The first mode is associated with the column space of $\mathbf{A}$ with dimensionality I_1. It holds $\mathbf{A}_{(1)} = \mathbf{A}$. The second mode is associated with the row space of $\mathbf{A}$ and has the dimensionality I_2. The mode-2 matrix of $\mathbf{A}$ equals the transpose of $\mathbf{A}$: $\mathbf{A}_{(2)} = \mathbf{A}^T$. The SVD $\mathbf{A} = \mathbf{U}\mathbf{S}\mathbf{V}^T$ can now be understood as a decomposition that orthogonalizes the column- and row-space of $\mathbf{A}$ via the matrices $\mathbf{U} \in \mathbb{R}^{I_1 \times J_1}$ and $\mathbf{V} \in \mathbb{R}^{I_2 \times J_2}$ and can be written in terms of mode-n products as $\mathbf{A} = \mathbf{S} \times_1 \mathbf{U} \times_2 \mathbf{V}$. Analogously, we can now define a decomposition of a general tensor which orthogonalizes the vector spaces associated with each mode:

Definition 3.7. (N-mode SVD)
Given a general tensor with $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_N}$ *with* $N > 2$ *the orthogonal column space of each mode-n matrix* $\mathbf{A}_{(n)}$ *can be represented by an unitary matrix* $\mathbf{U}_n$ *which can be computed, e.g. using the SVD of* $\mathbf{A}_{(n)}$*. Then the decomposition*

$$\mathcal{A} = \mathcal{Z} \times_1 \mathbf{U}_1 \times_2 \mathbf{U}_2 \dots \times_N \mathbf{U}_N$$

is called the N-mode SVD *of* $\mathcal{A}$*. The tensor* $\mathcal{Z}$*, which is given by*

$$\mathcal{Z} = \mathcal{A} \times_1 \mathbf{U}_1^T \times_2 \mathbf{U}_2^T \dots \times_N \mathbf{U}_N^T$$

governs the interaction between the modes and is termed core tensor.

The core tensor represents the analogue to the singular value matrix $\mathbf{S}$ in conventional SVD. Unfortunately, $\mathcal{Z}$ usually does not have a simple diagonal structure but is a full tensor. A so-called *rank*-$(R_1, R_2, \ldots, R_N)$ approximation $\tilde{\mathcal{A}}$ of $\mathcal{A}$, which means *rank*$(\tilde{\mathbf{A}}_{(n)}) = R_n \leq$ *rank*$(\mathbf{A}_{(n)})$ for $1 \leq n \leq N$, can now easily be achieved by truncating the modes $\mathbf{U}_n$ such that $\tilde{\mathbf{U}}_n = (\mathbf{u}_{n,i})_{1 \leq i \leq R_n}$ where $\mathbf{u}_{n,i}$ is the i-th column of $\mathbf{U}_n$. As already indicated this straight-forward way of reducing the dimensionality of a tensor is generally not optimal, i.e. it does not yield a global optimum of the approximation error $||\mathcal{A} - \tilde{\mathcal{A}}||$, but it works reasonably well in practice [VT04].

As proposed in [LMV00] the approximation can be further improved by using an ALS algorithm. The algorithm improves each mode matrix $\tilde{\mathbf{U}}_n$ iteratively by projecting the tensor onto all rank reduced modes except n: $\mathbf{M} = \left(\mathcal{A} \times_1 \tilde{\mathbf{U}}_1^T \times_2 \ldots \times_{n-1} \tilde{\mathbf{U}}_{n-1}^T \times_{n+1} \tilde{\mathbf{U}}_{n+1}^T \times_{n-2} \ldots \times_N \tilde{\mathbf{U}}_N^T\right)_{(n)}$. Then the matrix $\mathbf{M}$ is orthogonalized using SVD and $\tilde{\mathbf{U}}_n^{new}$ is set to the first R_n columns of the left singular matrix of $\mathbf{M}$. This step is repeated for all modes and iterated until convergence which can be determined for example by the amount of improvement in approximation quality between consecutive steps.

The storage requirements for a *rank*-$(R_1, R_2, \ldots, R_N)$ tensor approximation are $\mathcal{O}(\prod_n^N R_n)$ for the core tensor and $\mathcal{O}(\sum_n^N I_n R_n)$ for the mode matrices. While the significant overhead of the core tensor storage is not required in the matrix case, it still can be expected that the per-mode ranks R_i can be chosen relatively small. Furthermore, the mode dimensions I_n will be much smaller than the matrix dimensions which considerably reduces the storage requirements for the mode matrices compared to the Eigenvectors and weights in the standard SVD case.

Computing the HOSVD of a tensor requires decomposing N matrices where the n-th matrix has the dimensions $I_n \times \prod_{k \neq n}^N I_k$. Using an approximation algorithm linear in the number of matrix elements, we have to perform N times the work of a standard SVD which is $\mathcal{O}(\prod_{k=1}^N I_k R_n)$. It is often argued that tensor factorization has the advantage that the covariance matrices are small because the dimensions of individual modes are smaller compared to the dimensions of the 2D matrix arrangement and that, as a result, the tensor factorization could be computed faster. This is true if a full factorization is needed, but in practice, where only a *rank*-$(R_1, R_2, \ldots, R_N)$ approximation is needed there is no advantage of computing the full covariance compared to using a method like Online-SVD since computing the covariance matrix for a single mode n costs $\mathcal{O}(I_n \prod_{k=1}^N I_k)$ already. Furthermore, the core tensor also needs to be computed. This corresponds to projecting the data onto all orthogonalized subspaces where one single mode-n product costs $\mathcal{O}(\prod_{k=1}^n R_k \prod_{k=n}^N I_k)$. In the ALS algorithm such mode-n products have to be calculated N^2 times per iteration!

Reconstructing a single element from a tensor requires evaluating the follow-

ing mode-n product:

$$a_{i_1,i_2,\ldots,i_N} = \mathcal{Z} \times_1 \hat{\mathbf{u}}_{1,i_1}^T \times_2 \hat{\mathbf{u}}_{2,i_2}^T \ldots \times_N \hat{\mathbf{u}}_{N,i_N}^T \tag{3.4}$$

whereby $\hat{\mathbf{u}}_{n,i_n}$ is the i_n-th row vector of mode matrix $\mathbf{U}_n$. Since the core tensor $\mathcal{Z}$ is generally a full tensor, the reconstruction has costs of $\mathcal{O}(\prod_{n=1}^{N} R_n)$ which can be quite significant. In fact, compared to block-wise factorization approaches these costs are almost unacceptable in practical applications as will be shown in Section 4.4.

3.1.5 Alternative Methods

Reviewing and comparing the plethora of different matrix and tensor factorization techniques will be far beyond the scope of this thesis. Nevertheless, we will at least point at some important streams of research in matrix factorization which we consider are worth of further research and investigation in the field of BTF compression.

Non-Negative Matrix Factorization

The presented factorization techniques by now are all based on the classical singular value decomposition which guarantees the best rank-r approximation (per-block, per-mode) in the least-squares sense. Another important matrix factorization scheme is non-negative matrix factorization [LS00]. It computes a factorization of a non-negative matrix $\mathbf{V}$ into non-negative matrix factors $\mathbf{WH}$. These techniques achieved some attention around the beginning of this decade when graphics hardware was not able to process negative values. Prominent examples are the separable approximations for arbitrary BRDFs [KM99] or the non-negative factorization of surface light fields [CBCG02].

Recently, the technique had some sort of comeback in the form of a variant called *constraint factorization* which allows to enforce not only non-negativity but also other general constraints like sparsity and even domain-specific constraints like energy conservation if the respective factor is supposed to behave like a BRDF. This way factors can be computed which are especially useful for non-parametric editing as demonstrated in [LBAD+06].

Since non-negative factors are not a necessity for today's graphics hardware and the enforcement of constraints usually decreases approximation quality we will not cover the application of these technique for BTF compression here. However, it might be interesting for future work to investigate factorization techniques based on perceptually more meaningful error metrics than the standard L_2-norm.

Chained Matrix Factorization

In [SvBLD03] Suykens et al. proposed to factorize the per-texel reflectance functions of BTFs using a chain of factorizations. The idea here is to factorize the data in different parameterizations, whereas each factorization is meant to represent specific features favored by the particular parameterization. The main motivation of the approach was that, e.g. the half angle BRDF parameterization [Rus98] allows to transform the diagonal features resulting from the specular lobe into vertical features which can be factorized more efficiently. Generalizing this approach to the full BTF transport matrix would be also an interesting venue for future research.

Hierarchical Factorization

As already mentioned in the introduction of this thesis an important property of textures and appearance measurements in general is that they contain relevant features on various scales. A textile, for example, might consist of a repeating color pattern on a large scale caused by differently colored threads but there is also the weaving pattern of the threads themselves on a smaller scale etc. The important consequence for compression is that the large scale patterns should not be stored with the same resolution as the smaller scale patterns. This principle of scale-dependence lies behind multi-resolution approaches like hierarchical wavelet bases. In order to exploit the multi-scale structure of data in matrix factorization there are principally two different approaches.

The first approach is to transform the data into different frequency bands which are then factorized independently. Laplacian pyramids have been used in the literature [MCT+05] but wavelet transforms can be used in the same way. The other approach is to use hierarchical block matrix schemes like $\mathcal{H}$-matrices [Hac99]. Here the matrix is recursively subdivided into blocks and each block is approximated using a low-rank factorization whereas the recursion stops when the reconstruction error lies below a given threshold. This scheme can be easily generalized to tensors and has been applied to reflectance fields in [GTLL06]. A variant of this approach which keeps and approximates the residual matrix/tensor block on each level has been introduced in [WXC+08].

3.2 Summary

To summarize this background chapter about matrix factorization we collected the main properties of the presented techniques in Table 3.1. The insights that can be gained from this table are the following:

	SVD	Block-wise	
		Regular	Adaptive
Time	mnr	mnr_b	$t(p_n + p_m)nmr_b$
Storage	$(m+n)r$	$(mp_n + np_m)r_b$	
Reconstruct	r	r_b	
	Two-pass	Tensor factorization	
Time	$(mn + nm_b s)r_b$	$tN\left(\sum_n^N \prod_{k=1}^n R_k \prod_{k=n}^N I_k\right)$	
Storage	$mr_b + (n + p_m r_b)s$	$\sum_{n=1}^N I_n R_n + \prod_{n=1}^N R_n$	
Reconstruct	$r_b s$	$\prod_{n=1}^N R_n$	

Table 3.1: A general comparison of the presented matrix/tensor factorization schemes regarding their time complexity, storage requirements after rank reduction (not during computation!) and reconstruction cost for a single matrix element. All terms should be read in $\mathcal{O}$-notation. t denotes the number of iterations of the k-means and ALS methods respectively (typical values are $10 \leq t \leq 15$). The other variables are explained in the text.

Factorization Complexity Factorization costs are usually linear in the number of matrix elements times the truncated rank. For more sophisticated techniques like adaptive blocks or multi-linear tensor factorization additional factors like the number of iterations, the number of clusters or the number of modes affect the figures.

Storage Reduction Storage requirements are generally reduced to a sum of per-mode dimensions times additional factors like the truncated rank and the number of blocks. Here, the final performance of a method heavily depends on the per-mode or per-block dimensions respectively and on how low the rank within these factors can be chosen for a given error threshold. It is expected that in comparison to standard SVD more elaborate methods like tensor or block-wise factorization allow to lower the rank significantly, resulting in an improved compression ratio and/or cheaper reconstruction costs.

Reconstruction Complexity Reconstruction cost per element is an important issue since the techniques should be applicable for rendering applications, which requires almost random access at best possible speed. With respect to reconstruction cost, the factorization techniques presented here can be roughly divided into

two groups. On one side, there are the methods which require only the computation of a single inner product per reconstructed element (standard SVD and block-wise factorization). On the other side, there are the more elaborate methods which require the reconstruction of an additional subspace or the evaluation of mode-n products (two-pass and tensor factorization). The question that needs to be answered is, if the per-mode/subspace ranks can be chosen such that their product is comparable in size to the per-block rank.

CHAPTER 4

FACTORIZING THE BTF TRANSPORT MATRIX

In the previous chapter several general matrix factorization schemes as well as algorithms for their computation have been introduced. According to Table 4.1 except for the two-pass factorization variants of the mentioned techniques have already been applied to BTF data in the literature. In this chapter we will contribute a thorough comparison between these methods except two-pass factorization regarding their practical applicability in terms of fitting, storage and reconstruction cost.

The outcome of this chapter will be a translation of Table 3.1 into practical expressions depending on BTF sample parameters. This requires specifying the arrangement of the 6-dimensional BTF data in a 2D matrix and general tensor and how we want to deal with color. These topics will be discussed in Sections 4.1-4.3.

In Section 4.4 we will then perform the actual comparison in the form of several empirical experiments.

4.1 Matrix Data Arrangement

The discrete BTF representation introduced in Equation 2.15 can already be interpreted as a 2D matrix with $|K|$ rows and $|I^d|$ columns. Each row is a discrete reflectance function of a single pixel seen under a fixed view direction for varying light directions. Since $|K|$ is the number of pixels times the number of view directions it usually holds $|K| \gg |I^d|$ (e.g. $|K| = 256 \times 256 \times 151$ and $|I^d| = 151$) which is not quite adequate for effective factorization. Therefore, we will take a closer look at the index sets K and I^d. I^d indexes the illumination basis and is usually given by a set of l light directions:

$$I^d = \{(\theta_{i,0}, \phi_{i,0}), (\theta_{i,1}, \phi_{i,1}), \ldots, (\theta_{i,l}, \phi_{i,l})\}$$

K describes the discretization of the outgoing light-field and is usually given by the image or texture size $E = W \times H = \{1, 2, \ldots, w\} \times \{1, 2, \ldots, h\}$ and a

SVD	Block-wise		Two-pass	Tensor
	Regular	Adaptive		
[KMBK03]	[MPN+02]			[VT04]
[LHZ+04]	[SvBLD03][SSK03]	[MMK03]	X	[WWS+05]

Table 4.1: Matrix- and tensor factorization techniques published in the BTF compression literature.

set of v outgoing directions $O = \{(\theta_{o,0}, \phi_{o,0}), (\theta_{o,1}, \phi_{o,1}), \ldots, (\theta_{o,v}, \phi_{o,v})\}$ hence

$$K = E \times O = \{(1, 1, \theta_{o,0}, \phi_{o,0}), (1, 1, \theta_{o,1}, \phi_{o,1}), \ldots, (w, h, \theta_{o,v}, \phi_{o,v})\}.$$

Now $K \times I^d$ can be interpreted as a six-dimensional set of indices and $\mathbf{B}$ can be unrolled along each of these dimensions which will be exploited in the following section about tensor factorization. For the purpose of 2D matrix factorization the usually most balanced arrangement will be the one whose rows and columns vary by the spatial and angular indices respectively:

$$\hat{\mathbf{B}} := \mathbf{B}^{E \times (O \times I^d)} \tag{4.1}$$

The rows $\hat{\mathbf{b}}^{(x,y)}$ are called *4D reflectance functions* and can be interpreted as sampled per-pixel BRDFs [1]. The columns $\hat{\mathbf{b}}_{(o,i)}$ can be interpreted as images of the material for one lighting and viewing direction.

Let us now translate the figures from Table 3.1 into values related to $\hat{\mathbf{B}}$. We now have $m = |E| = w \cdot h$ as the number of rows and $n = |O \times I^d| = v \cdot l$ as the number of columns. The raw size of $\hat{\mathbf{B}}$ is hence size$(\hat{\mathbf{B}}) = w \cdot h \cdot v \cdot l$.

According to the tables on the right of Figures 2.5 and 2.6 values of $81 \leq v = l \leq 151$ are typical sampling rates for the the angular domain. In the spatial domain we have usually $64 \leq w = h \leq 512$ as typical texture sizes. Assuming a numerical accuracy of 16 bits (using for example the *half* datatype) the storage requirements for $\hat{\mathbf{B}}$ range from $2 \cdot 64^2 \cdot 81^2 \approx 54$ MB[2] up to $2 \cdot 512^2 \cdot 151^2 \approx 11.95$ GB.

Standard SVD allows to reduce these numbers to something between $2r \cdot (64^2 + 81^2) \approx r \cdot 21$ KB and $2r \cdot (512^2 + 151^2) \approx r \cdot 570$ KB. According to [KMBK03] we expect r to lie around 10^2–$2 \cdot 10^2$ which would result in 2-114 MB of compressed storage.

[1] Wong et al. [WHON97] proposed to call these functions *apparent BRDFs* because they have the structure of a BRDF but contain neighborhood effects (masking, shadows, interreflections etc.) that violate the BRDF principle.

[2] 1 MB $= 10^6$ bytes

If a blocking scheme with p_{wh} blocks along the spatial dimension and p_{vl} blocks along the angular dimension is used we have to calculate with memory requirements between $r_b p_{wh} p_{vl}(\frac{8}{p_{wh}} + \frac{13}{p_v l})$ KB and $r_b p_{wh} p_{vl}(\frac{524}{p_{wh}} + \frac{46}{p_v l})$ KB. We hope to reduce r_b by one order of magnitude compared to r but expect that around 50-100 blocks might be required. The number of blocks along spatial and angular dimension should be chosen depending on the sampling resolution along the respective dimensions.

Until now, only methods have been published which use per-column or per-row blocks exclusively as in [SSK03, MMK03] or a quite high number of regular blocks [MPN+02]. It can be expected that general adaptive block scheme based on co-clustering improves this state of the art.

4.2 Tensor Data Arrangement

The BTF measurement index set $W \times H \times O \times I^d$ gives rises to a natural 4-th order tensor which consists of row, column, outgoing and incoming direction modes:

$$\hat{\mathcal{B}} := \mathbf{B}^{W \times H \times O \times I^d} \in \mathbb{R}^{|W| \times |H| \times |O| \times |I^d|} \tag{4.2}$$

This arrangement has been proposed in [WWS+05]. They argued that using more modes (a 6-th order tensor seems to be the most natural arrangement for the 6D BTF) for the 2D spaces of in- and outgoing directions is not appropriate since there are too few samples along the modes of elevation and azimuthal angle. Indeed, our test samples consist of only 81-151 samples for the whole 2D spaces of in- and outgoing directions. On the other hand it is also possible to use less then four modes as it was proposed by [VT04] in order to reduce run-time reconstruction cost:

$$\hat{\mathcal{B}}_{\mathsf{T}} := \mathbf{B}^{E \times O \times I^d} \in \mathbb{R}^{|E| \times |O| \times |I^d|} \tag{4.3}$$

This particular arrangement is also known as Tensor-Textures.

Of course, as above the raw size of the tensor is $\hat{\mathcal{B}} = w \cdot h \cdot v \cdot l$. The expected compression ratio is quite more difficult to judge without experiments because a custom rank can be chosen for each mode. Indeed, with the chosen range of sampling rates we have storage requirements from $2 \cdot (64 \cdot R_w + 64 \cdot R_h + 81 \cdot R_v + 81 \cdot R_l + R_w R_h R_v R_l)$ bytes up to $2 \cdot (512 \cdot R_w + 512 \cdot R_h + 151 \cdot R_v + 151 \cdot R_l + R_w R_h R_v R_l)$ bytes.

In their original paper Wong et al. have chosen $R_w = w/2$, $R_h = h/2$ and R_v, R_l between 20 and 30. This would lead to storage requirements from 819 KB up to 118 MB for the core tensor alone which is comparable to the standard SVD case. Unfortunately, in comparison to the matrix based approaches the reconstruction cost would be significantly higher in this case. Alternatively, the

spatial dimension could already be reconstructed in a preprocess which would lead to the TensorTexture representation $\hat{\mathcal{B}}_\mathsf{T}$. The reconstruction cost then would be comparable to the standard SVD case with the only difference that a "*strategic dimensionality reduction*" [VT04, p.4] becomes possible which allows to weight the importance of view and lighting variation. The motivation is that a reconstruction which preserves more variance in the view mode can be perceptually better than a SVD reconstruction even if its RMS error is higher.

4.3 Dealing with Color

Up to now we have neglected the spectral dependency of the discrete BTF representation. In practice, measurements of light transport are performed for a set Λ of color channels or spectral bands λ which means a complete BTF matrix $\hat{\mathbf{B}}_\lambda$ is required for each $\lambda \in \Lambda$. Hence the index set for every single measurement pixel will be $\Lambda \times E \times O \times I^d$. Since our test case BTFs have been measured using the standard RGB color representation we have $\Lambda = \{\lambda_r, \lambda_g, \lambda_b\}$. One of the simplest solutions would be to unroll this additional dimension into the rows or columns of the matrix[3]:

$$\hat{\mathbf{B}}_\Lambda := [\hat{\mathbf{b}}_{1,\lambda_r}\ \hat{\mathbf{b}}_{1,\lambda_g}\ \hat{\mathbf{b}}_{1,\lambda_b}\ \ldots \hat{\mathbf{b}}_{|E|,\lambda_r}\ \hat{\mathbf{b}}_{|E|,\lambda_g}\ \hat{\mathbf{b}}_{|E|,\lambda_b}] \tag{4.4}$$

Now the rows of this matrix contain rgb-reflectance functions. This way, the correlation between the different color bands is not exploited optimally. Alternatively, we could already anticipate the presented more sophisticated factorization techniques here. For example the color bands could be interpreted as an additional mode in tensor factorization. Nevertheless, this seems a bit like using a sledgehammer to crack a nut because there are usually only three color bands which are not amenable to rank reduction. Therefore, our approach will be simply to apply a de-correlating transform $T_C : \mathbb{R}^{|\Lambda|} \rightarrow \mathbb{R}^{|\Lambda|}$ to each color triple $(b_{\lambda_r}, b_{\lambda_g}, b_{\lambda_b})$:

$$(b_{c_0}, b_{c_1}, b_{c_2})^T = T_C\left((b_{\lambda_r}, b_{\lambda_g}, b_{\lambda_b})^T\right)$$

We prefer to use PCA as proposed in [HB95] in order to find a suitable (affine-linear) T_C but alternatively also non-linear color-space transforms like CIELab can be used as proposed in [GMSK08].

Afterwards, we assemble the BTF matrices $\hat{\mathbf{B}}_{c_i}$ per transformed color band c_i and factorize each matrix independently, i.e. we apply a simple block scheme on top of the actual factorization. Since most materials show a great deal of correlation between color bands most of the energy concentrates in one or two of the transformed color channels which allows to compress the less important channels

[3] We used here the well-known Matlab© notation for concatenating column vectors to a matrix

much more aggressively (cf. Figure 4.2). For rendering, the inverse transform T_C^{-1} has to be applied.

If not indicated otherwise, in the following the factorization algorithms will be always applied per single color channel matrix $\hat{\mathbf{B}}_{c_i}$ or tensor $\hat{\mathcal{B}}_{c_i}$.

4.4 Empirical Comparison

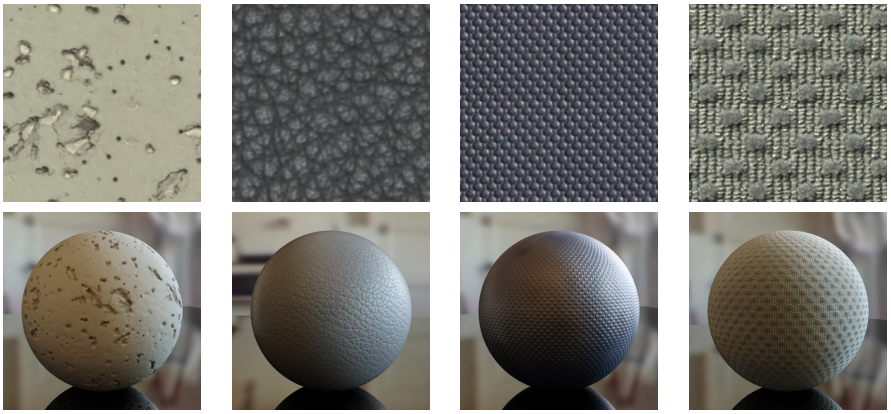
	PLASTER	LEATHER	PLASTIC	CUSHION
				
$\lvert O \times I^d \rvert$	81x81	81x81	151x151	151x151
$\lvert W \times H \rvert$	256x256	256x256	128x124	120x129
Raw	2,58 GB	2,58 GB	2,17 GB	2,18 GB
BZip	834 MB	881 MB	1,48 GB	1,26 GB

Figure 4.1: The set of materials used in the comparative study. The materials are stored with 16-bit precision per color-channel. Since the PLASTER and LEATHER materials have been acquired with 12-bit precision only, entropy compression using the BZip-algorithm reduces storage requirements already by 60%.

While Table 3.1 already gives the main properties of the reviewed factorization techniques their practical performance mainly depends on the selection of the various rank parameters, as there are matrix rank r, per-block rank r_b and per-mode rank R_n, and the row and column partitioning parameters p_m and p_n. Therefore, in this section the various techniques will be applied to a few real world BTF samples in order to get a notion how these parameters have to be chosen for satisfying results in practice. The four different BTFs of varying complexity used here are depicted in Figure 4.1. The first two materials (PLASTER, LEATHER) were captured with the gonioreflectometer-like setup from the University of Bonn (see

Section 2.6.3). The other materials (PLASTIC, CUSHION) were measured with the multi-camera dome also from Bonn.

Each of the samples will be compressed with the following matrix factorization techniques:

- SVD/global factorization
- fixed and adaptive block-wise factorization with one and two-way clustering
- tensor factorization with four modes (image rows, image columns, view direction, light direction) using the Tucker ALS algorithm

Each technique is applied with different parameters such that a practical range of compression ratios is covered. All data values will be stored as half precision 16-bit floats according to the proposed revised version of the IEEE 754 standard. Different quantization techniques could be used here but such an investigation would be out of the scope of this thesis. Reconstruction error and cost are computed for each parameter set as follows:

The average root-mean-square error per reflectance function

$$e_{RMS}(\mathbf{A}, \tilde{\mathbf{A}}) = \frac{1}{m} \sum_{i}^{m} \sqrt{\sum_{j}^{n} \frac{|a_{ij} - \tilde{a}_{ij}|^2}{n}}$$

is computed and then plotted against the compression ratio. More sophisticated error metrics might be used in future experiments but from our experience the RMS error is still a good compromise of expressivity and computational complexity.

The reconstruction cost for each technique and each parameter set is computed empirically by rendering an image of a simple object covered with the BTF several times using a simple path-tracer[4] and also plotted against the compression ratio. Both curves could usually easily be identified in the plots since the RMSE increases from left to right while the reconstruction cost decreases. We plot the rendering time subtracted by the time required for rendering the same scene without any shading calculations. This way a more practical statement can be made compared to the simple formulas for the reconstruction cost shown in Table 3.1. Apart from the mere cost of reconstructing the discrete matrix elements the real reconstruction costs contain also the additional costs for interpolation and color reconstruction and are influenced by difficult to estimate but significant factors like memory locality.

[4]The path-tracer has been written in C++ by the author of this thesis. It supports lighting with environment maps and has been used to render most of the images in this work.

All computations were performed on a standard PC system with a 2.4 GHz Intel(R) Core(TM) 2 CPU and 4 GB RAM. All algorithms have been implemented both in C++ and in MATLAB ©[5]. For the C++ version we exploited the ublas[6] and CLAPACK[7] packages for linear algebra computations without multi-threading, i.e. only a single core was used.

4.4.1 Standard SVD

As suggested in Section 4.3 we apply SVD to the decorrelated color channels $\hat{\mathbf{B}}_{c_i}$ independently:

$$\hat{\mathbf{B}}_{c_i} \approx \mathbf{T}_{c_i}\mathbf{S}_{c_i}\mathbf{B}_{c_i}^T \tag{4.5}$$

Here $\mathbf{T}_{c_i} \in \mathbb{R}^{|E|\times r_i}$ is the matrix of per-channel *Eigen-textures* and $\mathbf{B}_{c_i} \in \mathbb{R}^{|O\times I^d|\times r_i}$ is the matrix of per-channel *Eigen-reflectance functions*.

Computation Time

As already mentioned in Chapter 3 we used EM-PCA [Row98] whose complexity scales linearly with the number of matrix entries and the desired rank. The original paper proposes two different implementations of the algorithm: an in-core version which requires to hold the whole data matrix in main memory and an online version suitable for out-of-core computations which processes only a single column at a time and thus needs to hold only the eigenspace in memory.

It turned out that in practice the in-core version of the algorithm should be preferred since it is much faster due to better memory locality. In fact, using Sam Roweis' original implementation the online version is about 40 times slower than the in-core version. Using the 64-Bit version of MATLAB, which allows to factorize our test BTFs with a chosen reduced rank of $r = 256$ completely in-core, the factorization took about 20 minutes on our test machine while the out-of-core algorithm required about 14 hours. We implemented also an optimized C++ version of the online algorithm but it reduced computation times only by about 50 percent which suggests that the in-core algorithm should be used whenever possible.

Storage Requirements

Given a targeted compression ratio the per-channel rank r_i is simply determined by sorting the set of all per-channels singular values $\{\sigma_{c_i,j}\}$ according to their

[5] MATLAB version 7.4.0. (2007a) Natick, Massachusetts: The MathWorks Inc., 2007

[6] `http://www.boost.org`

[7] `http://www.netlib.org/clapack/`

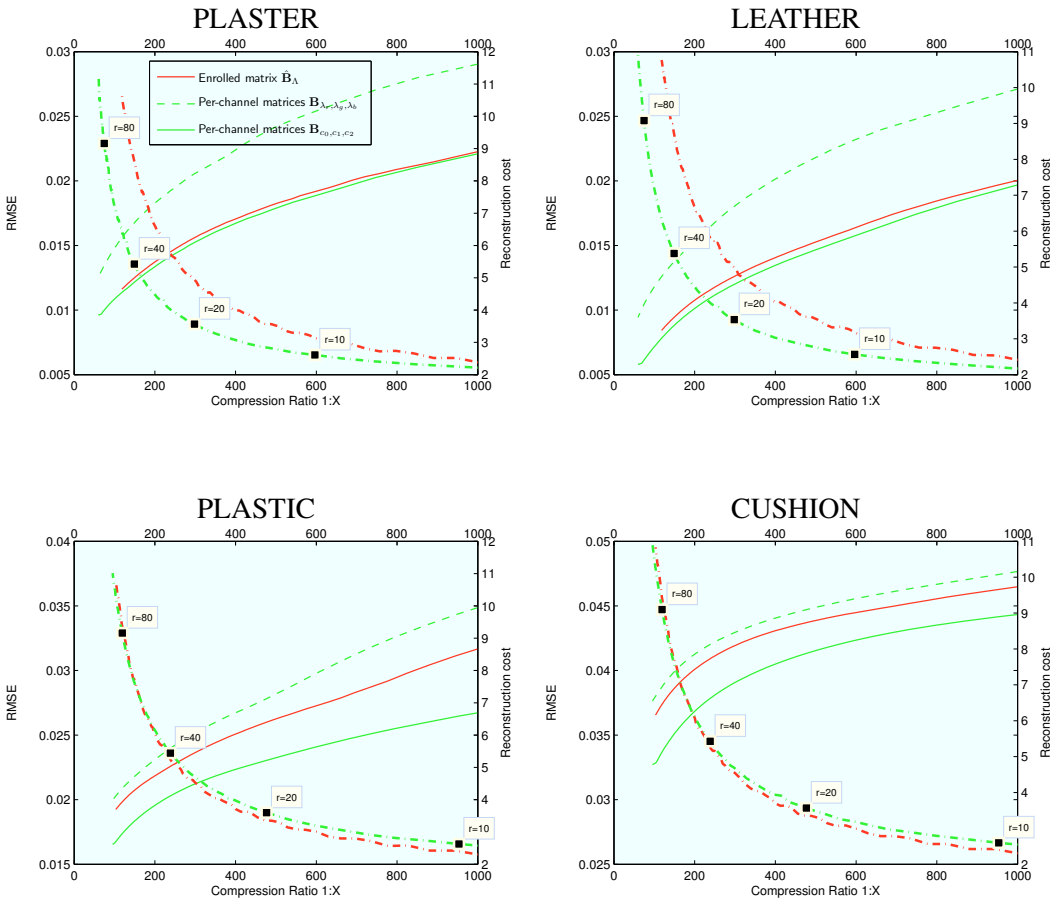


Figure 4.2: RMSE and reconstruction cost (in seconds, dash-dotted lines) of standard SVD for a given compression ratio (green stands for per-channel factorization and red for the unrolled matrix). Generally, the RMSE increases while the reconstruction cost decreases from left to right which helps identifying the curves. Factorizing $\hat{\mathbf{B}}_\Lambda$ requires less storage per component but more components are needed to achieve a RMSE similar to the decorrelated per-channel factorization. Per-channel factorization without color decorrelation is inefficient.

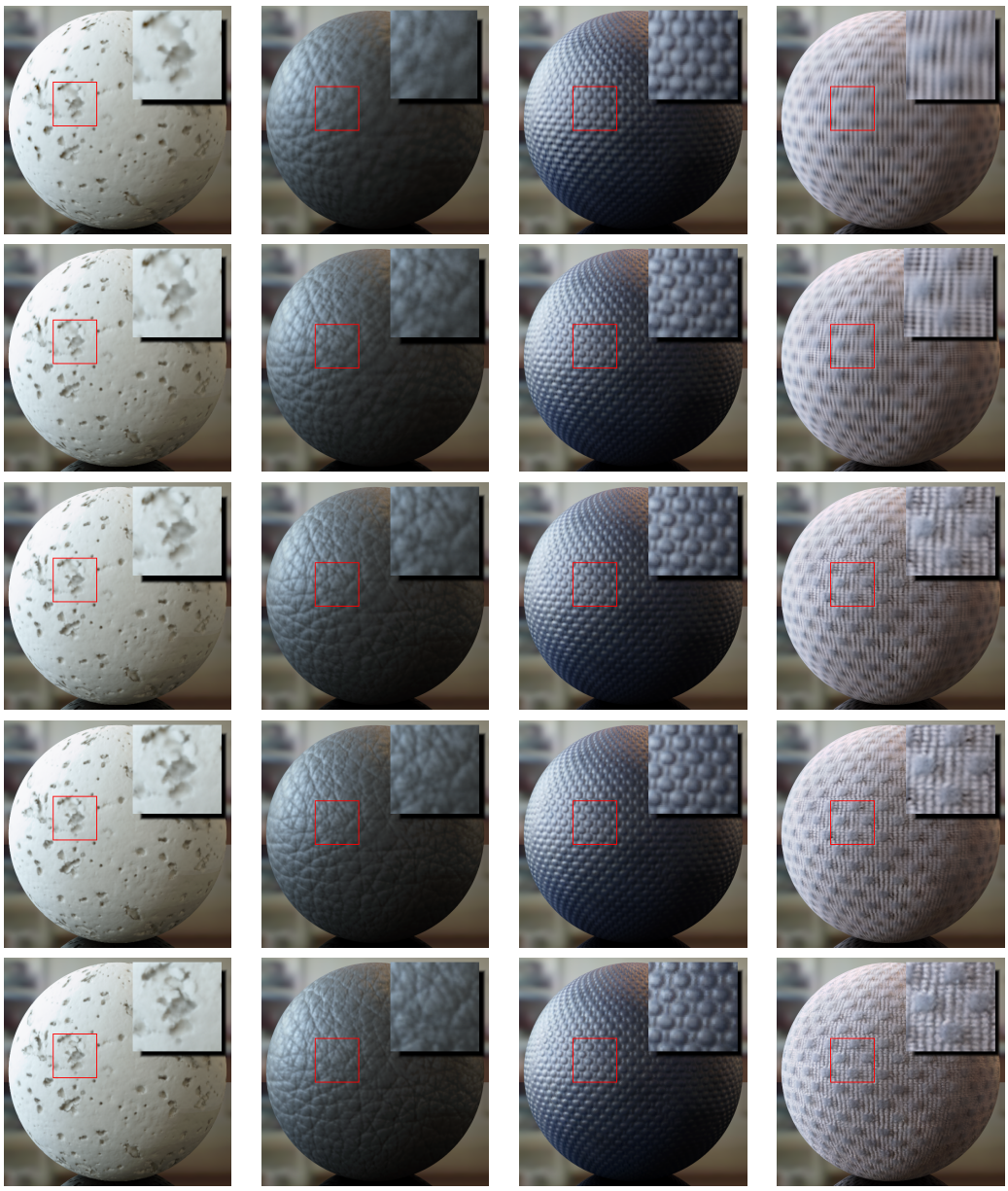


Figure 4.3: Renderings from standard SVD compression for $r = 10, 20, 40, 80$ (from top to bottom). The bottom row shows the reference images.

magnitude and increasing the corresponding channel rank starting from the biggest singular value until the storage budget is reached. The benefit gained from de-correlating color channels can be easily seen by comparing against factorizing the original color channel matrices $\hat{\mathbf{B}}_{\lambda_r,\lambda_g,\lambda_b}$ and the completely unrolled 2D-matrix $\hat{\mathbf{B}}_{\Lambda}$ as illustrated in Figure 4.2.

In order to get an impression what the RMS error tells about the visual quality of the reconstruction we rendered a sphere covered with each material for truncated ranks of $r = 10, 20, 40, 80$. The corresponding compression ratios are marked in the diagrams in Figure 4.2. It can be noted that for the simpler materials like PLASTER and PLASTIC already 20 components deliver a satisfying reconstruction which corresponds to compression ratios of 1:300 or 1:477 ($\approx$ 8 and 4.5 MB) respectively. The LEATHER material requires at least 40 components (1:150, 16 MB) while for CUSHION even the reconstruction with 80 components (1:120, 18 MB) still shows some visible differences to the reference image.

Reconstruction Cost

The reconstruction cost scales roughly linearly with the stored rank. For $r = 10$ the pure reconstruction cost are 2.5 seconds, for $r = 40$ they are around 5 seconds and for $r = 80$ about 10 seconds.

4.4.2 Block-wise SVD

Factorizing color channels independently already corresponds to a simple regular block scheme. Improvements in terms of compression ratio and reconstruction cost were possible because a de-correlating transform was applied which reduced the variance of the approximating matrices and introduced only a minor overhead during reconstruction. Now the question arises if BTF factorization benefits in a similar way from more sophisticated block schemes. According to Section 3.1.2 we can choose to partition the row space of $\hat{\mathbf{B}}$ which corresponds to the BTFs angular domain or the column space which means to partition the spatial domain or both of them simultaneously. In all theses cases the blocks can be chosen to be regular or general tensor product blocks adapted to the data using a clustering algorithm like local PCA.

Computation Time

An advantage of block-wise SVD is that the matrix blocks usually fit into main memory which means that no out-of-core management despite the assembly of blocks is required and the in-core version of EM-PCA can be used. Furthermore, factorization costs are linear in r_b and usually $r_b < r$. For example, we used

$r_b = 48$ as maximum per-block rank in our experiments compared to $r = 256$ for the global factorization. As a result the complete block-wise factorization of our test data matrices takes only about three to four minutes per material.

Using adaptive partitioning with simple k-means (L2-error) requires two to three minutes per iteration (cluster assignment and mean computation) which results in about 30 minutes total computation time if ten iterations are used. Using the reconstruction error as clustering metric (local-PCA) requires to factorize the data in each iteration which means the full factorization time ($\approx$ 3 minutes) has to be multiplied by the number of iterations. Furthermore, evaluating the reconstruction error cluster metric is more expensive compared to simple k-means since each data vector has to be projected into each cluster subspace. Fortunately, from our experience after about five iterations the improvement in reconstruction error is negligible which means that we end up with computation times around one hour per material which is only three times slower than performing a global factorization of the whole matrix. These costs can be usually reduced further by using only a subset of the data (about 30-50% can be used without significantly affecting the results) during the clustering step.

Storage Requirements

Choosing an appropriate number of blocks is an interesting question and difficult to answer in practice. Therefore, we make a pragmatic decision here and create the following test cases which will be performed with regular and adaptive blocks:

1. Spatial partition using $p_m = 30$ partitions
2. Angular partition using $p_n = 30$ partitions
3. Spatial and angular partition such that the resulting matrix blocks are approximately quadratic.

Case 1 corresponds to the Local-PCA compression (adaptive spatial clustering using the reconstruction as cluster error metric) we published in [MMK03] and it is especially suited for spatially large BTFs with a sparse angular sampling because the storage requirements are dominated by the number of partitions times the number of angular samples. Case 2 is a variant of Sattler et al. [SSK03] where a fixed number of partitions equaling the number of sampled view directions was used. Since the storage requirements are then dominated by the number of view directions times the spatial extent of the BTF this method is only practical for small BTFs with a sparse sampling of views. The last case combines the two and thereby allows to adapt the size of matrix blocks to the spatial and angular sampling rates.

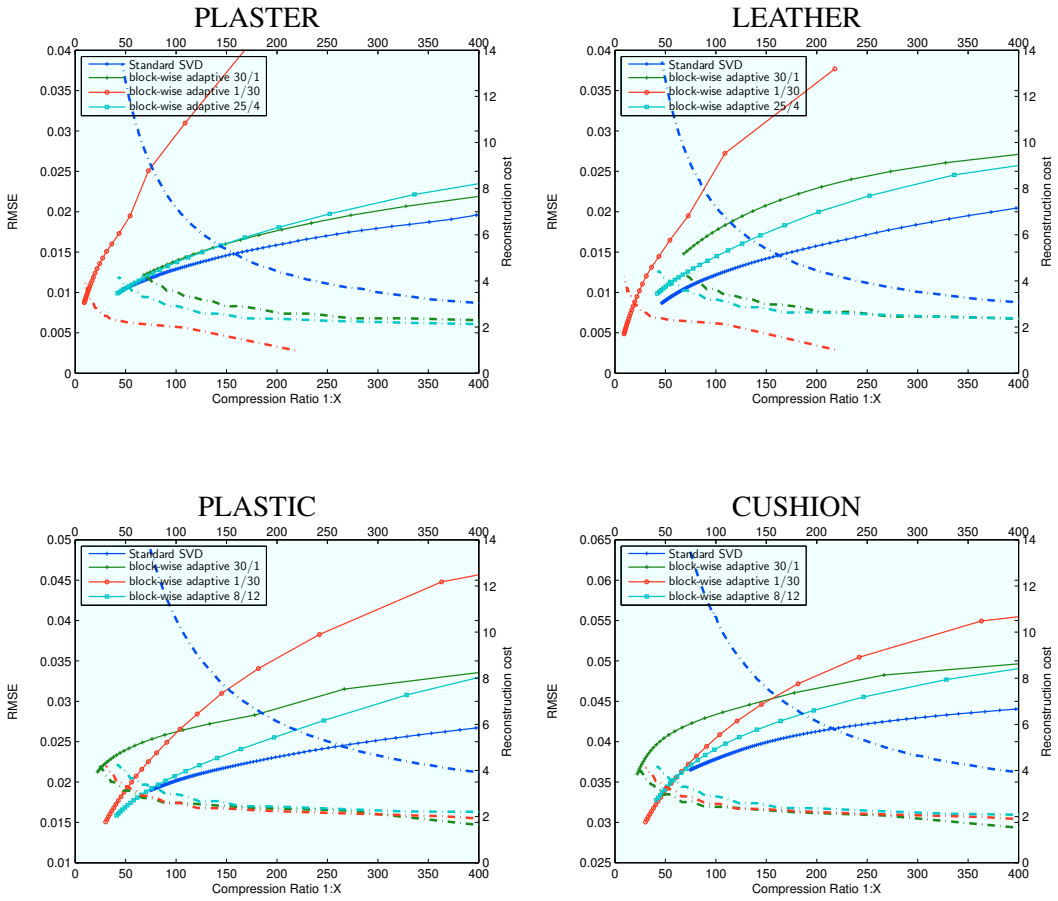
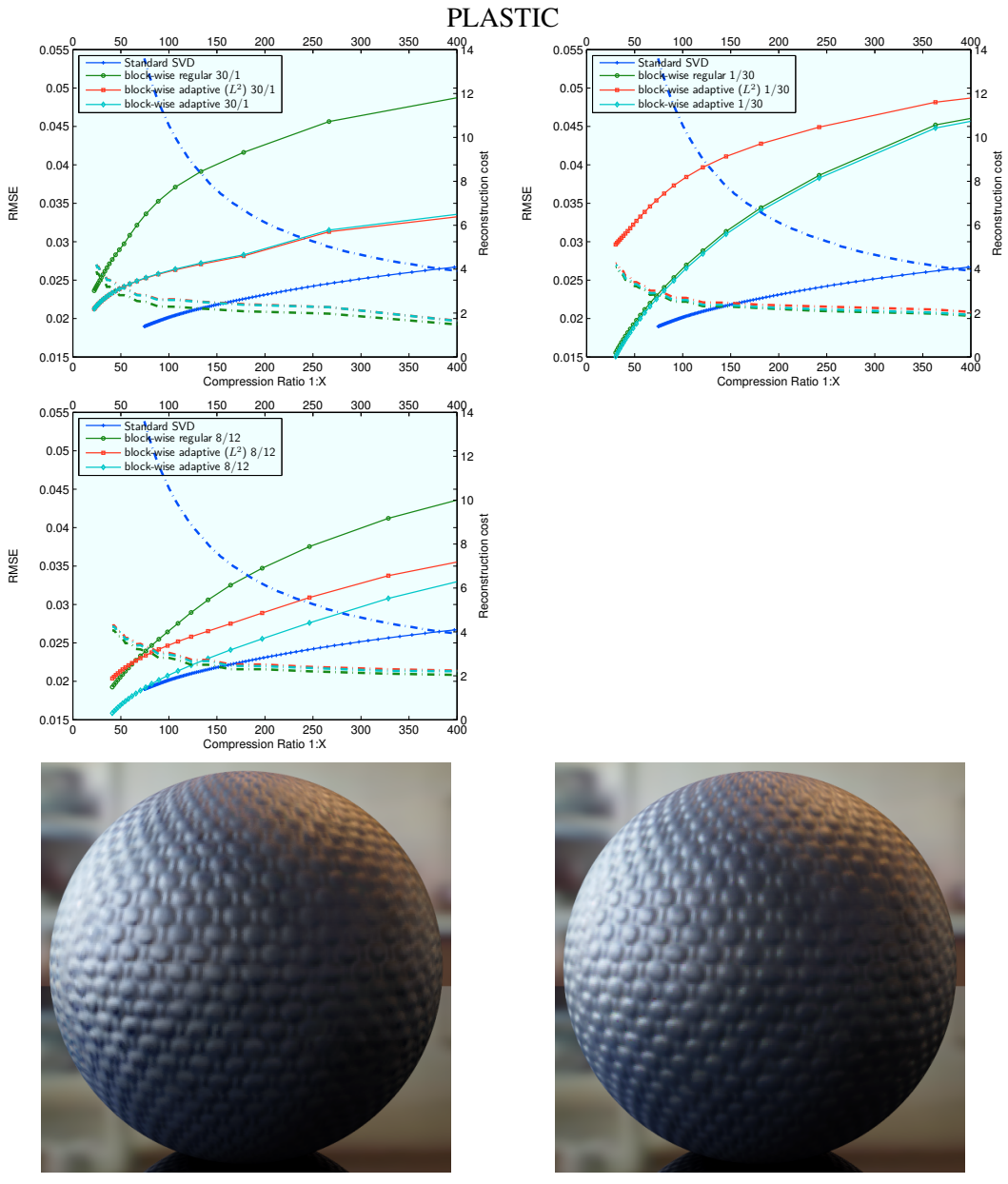


Figure 4.4: RMSE and reconstruction cost of adaptive block-wise factorization for a given compression ratio. Using general blocks, i.e. partitioning both rows and columns performs best and comes close to the compression ratio/RMSE performance of standard SVD but with significantly reduced reconstruction cost.



block-wise regular 8/12, RMSE=0.031

block-wise adaptive 8/12, RMSE=0.023

Figure 4.5: Comparing regular blocks with adaptive blocks using L^2-norm and reconstruction error as cluster error metric. The L^2 metric performs surprisingly bad for column clustering (top right). For row clustering there is almost no difference between the L^2-norm and the reconstruction error (top left). Co-clustering of rows and columns benefits from using the reconstruction error as cluster metric. Bottom row: the visual difference between regular (left) and adaptive (right) blocks for $r_b = 6$ is clearly notable as blurred highlights in the left image.

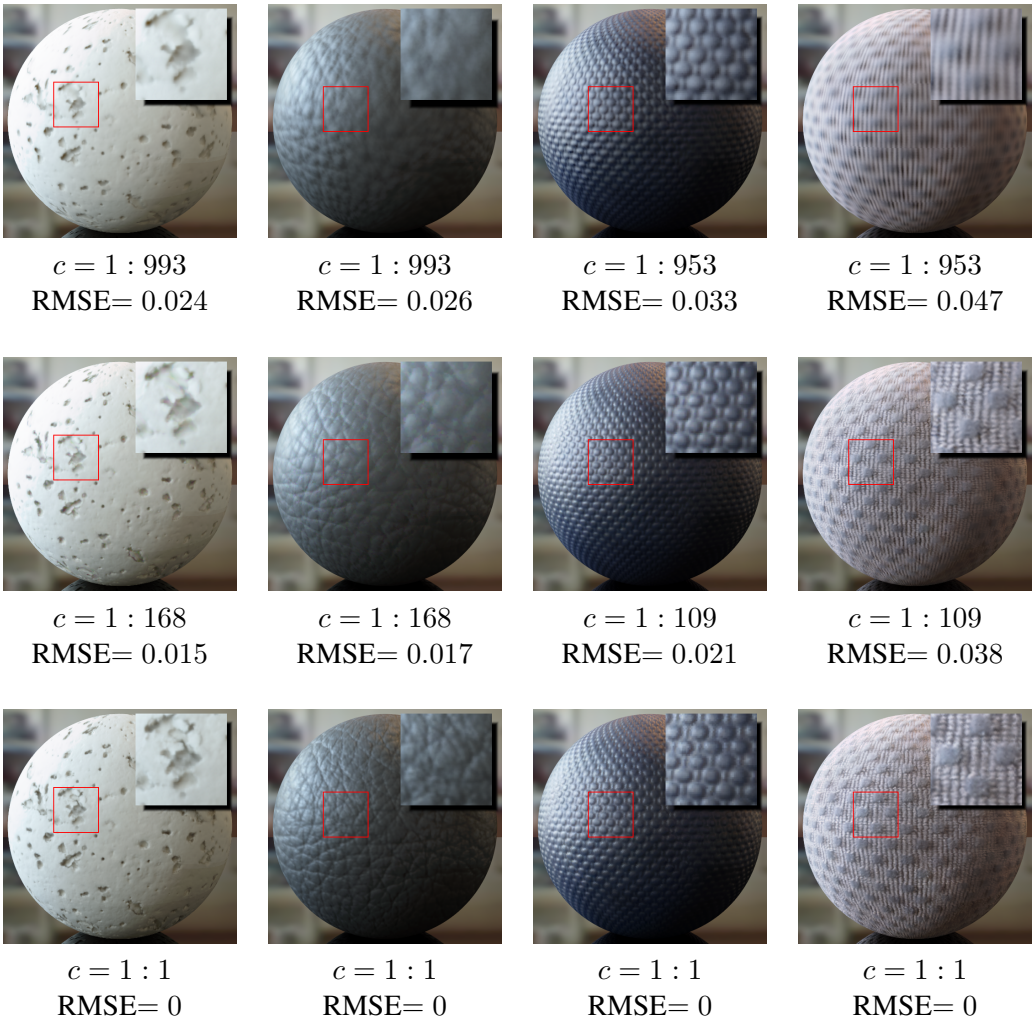


Figure 4.6: Comparing global (first row) and block-wise factorization using adaptive partitioning of both rows and columns (second row) with similar reconstruction costs. We have chosen rank $r, r_b = 6$ for the the PLASTER and LEATHER materials and $r = 10, r_b = 9$ for the PLASTIC and CUSHION materials. The improvement in reconstruction quality for the same (empirical) reconstruction cost is most visible for the complex materials LEATHER, PLASTIC and CUSHION. The bottom row shows reference images.

Figure 4.4 shows the experimental results of the three test cases for each of the test materials using adaptive blocks with the reconstruction error as cluster error metric. It can be seen that especially in the case of a *thin* matrix with uneven row and column dimensions (e.g. the PLASTER and LEATHER materials consist of $m = 65536$ rows and $n = 6561$ columns) it is quite inefficient to partition only along the smaller dimension. In fact for almost all inspected compression ratios simultaneous partitioning of both rows and columns performs best in terms of RMSE. For compression ratios around 1:100 the two-way partitioning even comes close to the standard SVD but with almost only one third of the reconstruction cost.

Regular vs. Adaptive Blocks As mentioned above the use of adaptive blocks requires significant computational costs during the fitting stage since the whole data has to be factorized during each iteration. Is it worth the effort in comparison to using only regular blocks or only the L^2-norm?

Figure 4.5 shows the test results for the PLASTIC material. The results for the other materials were qualitatively similar. It can be observed that adaptive clustering results in significant improvements for row partitioning, i.e. spatial clustering of reflectance functions. These improvements are also visually significant. Interestingly, this is not the case for column partitioning which corresponds to the clustering of images. Here, clustering based on the reconstruction error (local PCA) improves the error only marginally and simple k-means clustering even increases the error quite significantly compared to just using regular blocks (top right plot in Figure 4.5). In this case the L^2 norm is obviously not a suitable error metric. These results suggest the following practical approach which offers the best compromise between reconstruction quality and fitting cost: adaptive row partitioning using simple k-means and regular column blocks.

Reconstruction Cost

As before Figure 4.4 shows that the reconstruction costs scale roughly linearly with the stored rank. If the points of similar rank from the different block schemes are connected an almost perfectly horizontal line is obtained. In our example for $r = 16$ it is $c \approx 4s$ and for $r = 6$ we have $c \approx 2.3s$ which means the effect of improved cache locality due to a smaller memory footprint is not quite significant here. Nevertheless, as can been seen in Figure 4.5 the regular block schemes are always slightly faster than the adaptive block schemes. The reason is that the adaptive block schemes increase the probability of adjacent rows and columns to lie in different blocks which in turn increases the number of block changes during rendering. Apparently, such a block change usually implies a big hop in memory which is not desirable in terms of cache locality. This effect might become even

more significant for modern highly parallelized GPUs which are more sensible to cache misses and the available memory bandwidth than general purpose CPUs.

4.4.3 Tensor Factorization

Computation Costs

As suggested in [WWS+05] we used the Tucker-ALS algorithm [LMV00] for computing the tensor factorization. Since according to Section 4.2 there are $N = 4$ modes we need to factorize the complete data four times per iteration. For each factorization we need to project the data onto each of the other three modes consecutively which dominates the runtime cost of the method since for our datasets the maximum per-mode dimension is 256 (for the PLASTER and LEATHER materials) and the factorization of the flattened projected tensor can usually be performed in-core. The most costly projection is the one onto the first mode which requires $\mathcal{O}(I_{i_0} R_{i_0} I_{i_1} I_{i_2} I_{i_3})$ operations which equals $\mathcal{O}(mn R_{i_0})$. The runtime decreases for each consecutive mode but, e.g. the second mode still requires $\mathcal{O}(R_{i_0} R_{i_1} I_{i_1} I_{i_2} I_{i_3})$ operations which is still significant since in our examples usually $R_{i_0} = \frac{1}{2} I_{i_0}$ (the image row and column rank is chosen as half the pixel resolution). Therefore, the workload of the tensor factorization is roughly at least six times higher than the factorization of the corresponding matrix using, e.g. EM-PCA assuming that a comparable number of iterations is used. We employed the MATLAB tensor toolbox by Bader et al. [BK07] for our experiments which confirmed these theoretical thoughts.

In our experiments the ALS algorithm converged usually after about 10-15 iterations which is the about same of number iterations required for EM-PCA. Due to the complex memory access patterns which occur while projecting onto the different modes the expected factor of six compared to EM-PCA turned out to be rather factor of ten in practice which corresponds to computation times of about three hours for our test materials. During our experiments we also implemented the Tucker ALS algorithm ourselves in C++ and we discovered that an optimization for memory locality is extremely important since our unoptimized C++ code performed nearly 15 times slower than the MATLAB implementation.

Storage Requirements

The main motivation behind the use of Tensor Factorization is the hope that redundancies not represented by the standard co-variance matrix could be exploited to achieve improved compression ratios. In the case of BTF compression this corresponds to correlating not only the images or the reflectance functions respectively, but also the rows and columns of the images and the view- and and light

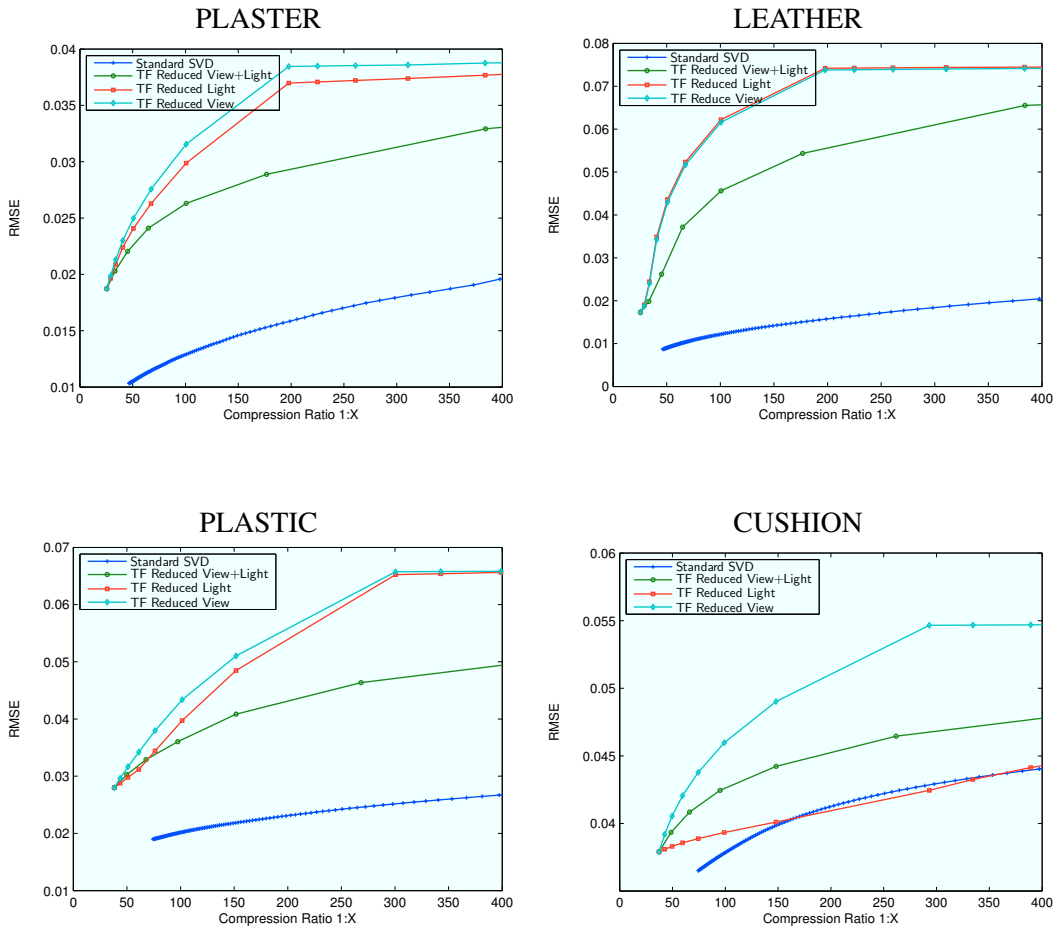


Figure 4.7: RMSE of tensor factorization compared to standard SVD. We used always $R_0 = 0.5 \cdot W$ and $R_1 = 0.5 \cdot H$ for the image row and column ranks. The maximum ranks R_2 and R_3 for the view- and light direction modes where set to 32 for PLASTER and LEATHER and to 48 for PLASTIC and CUSHION. We did not plot the reconstruction costs here, because they are about three orders of magnitude higher compared to standard matrix factorization.

variation of the reflectance functions. We can interpret this approach as a way to detect additional redundancies present in the classical SVD basis. According to Equation 4.5 this basis consists of Eigen-textures and Eigen-reflectance-functions which might exhibit significant self-similarity.

Indeed, Wang et al. reported in [WWS+05] an improved compression ratio for the same RMSE compared to SVD for some experiments. As in their original paper we set the image row and column ranks R_0, R_1 to half of the image resolution in our experiments. For the view- and light direction modes we kept a maximum of 32 (PLASTER, LEATHER) or 48 (PLASTIC, CUSHION) dimensions respectively and also performed a *strategic* dimensionality reduction as proposed by Vasilescu et al. [VT04] which means storing a different number of dimensions for the view- and light direction modes.

The results are plotted in Figure 4.7 and they show that we could confirm Wang's results only partly. In fact, except for the CUSHION material the RMSE is always significantly higher. This is particular evident if the correlation between image rows and columns is low as in the case of the LEATHER and PLASTER materials. Since the PLASTIC and CUSHION materials are spatially more structured and contain repeating elements the method performs better here but still mostly inferior compared to standard SVD. These findings are also confirmed by the rendered images shown in Figure 4.8. Especially for the LEATHER and PLASTIC materials and in all cases of view mode reduction (small R_2) the inferior RMSE is expressed by visually significant artifacts like spatial blurring and incorrect reflections.

Reconstruction Cost

Reconstruction cost is the main drawback of tensor factorization because rendering requires the reconstruction of single pixels which in turn requires the evaluation of Equation 3.4 which has complexity $\mathcal{O}(\prod_n^N R_n)$. In contrast, reconstruction from matrix factorization has only complexity $\mathcal{O}(r)$ or $\mathcal{O}(r_b)$ respectively. In our experiments these costs render the method literally impractical since we have, e.g. $20 \leq r \leq 80$ for SVD but $64 \leq R_{0,1} \leq 128$ and $30 \leq R_{2,3} \leq 48$. For example the CUSHION material requires $R_0 = 64, R_1 = 64, R_2 = 48, R_3 = 30$ meaning $\prod_n^4 R_n = 5.898.240$ to achieve an RMSE comparable to SVD with $r = 80$. In practice, we experienced about 2000-3000 times higher running times[8] for rendering the test images with tensor factorization shown in Figure 4.8 which corresponds to reconstruction times in the range of hours compared to seconds! In order to make the rendering feasible at all we pre-multiplied the image basis which yields the TensorTexture representation. This way the rendering times are

[8]For this reason we did not plot the reconstruction costs into the diagrams

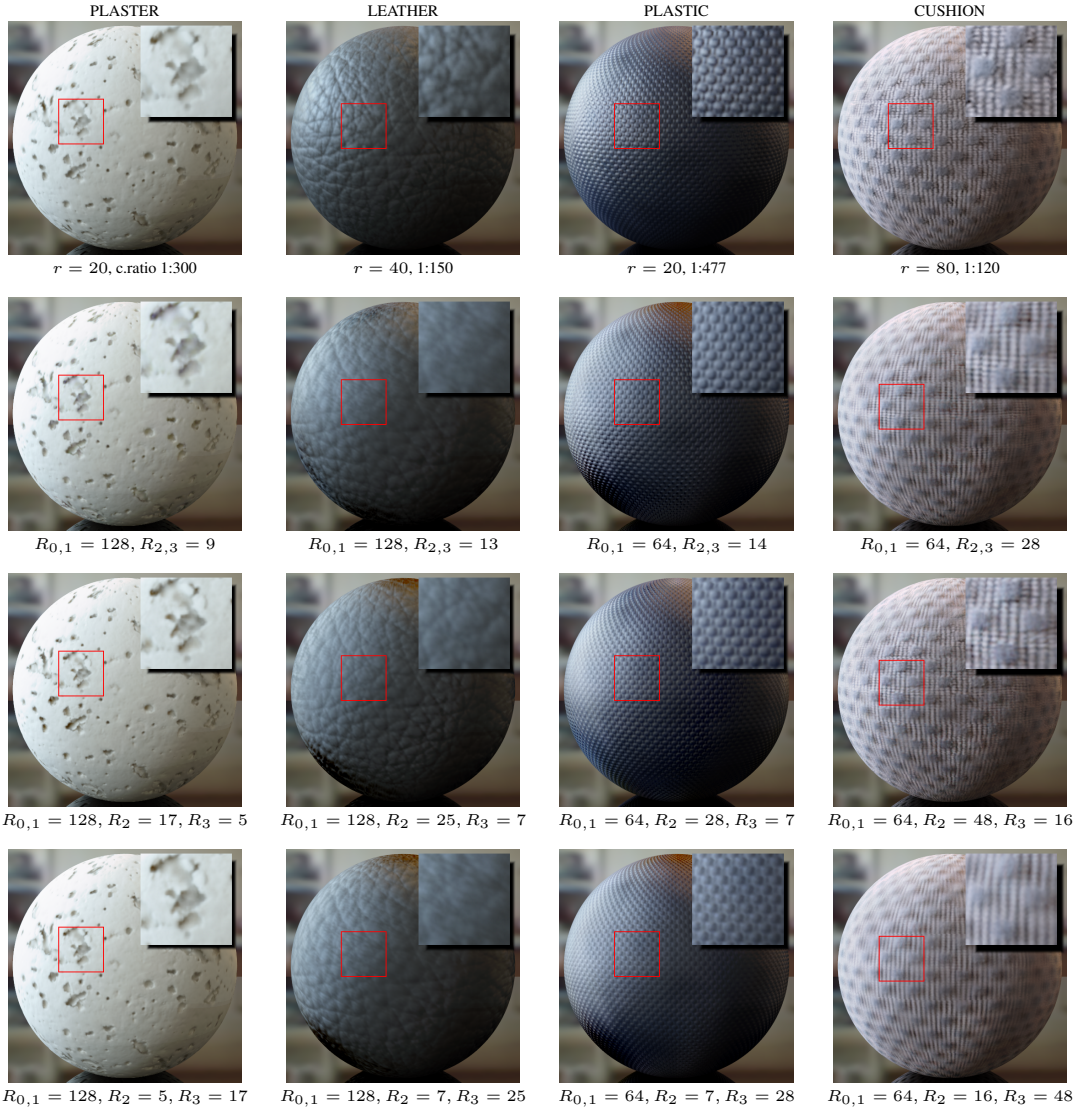


Figure 4.8: Comparing standard SVD and tensor factorization with *strategic dimensionality reduction* (different values for light/view modes R_2 and R_3) using the same per-material compression ratio. For the PLASTER material the differences are hardly noticeable. For the LEATHER and PLASTIC materials the higher RMSE of tensor factorization is visually significant especially in the case of reduced view mode (R_2) which leads to spatial blurring. Keeping the same rank for view and light modes performs best here. For the CUSHION material the RMSE and visual impression are comparable in the case of reduced R_3 (light mode). Please note also the significant differences in rendering time (see text).

	RMSE	SVD	Block-wise		TF
			Reg.	Adapt.	
Fitting Time		20-30 m	3-4 m	60-120 m	2-3 h
PLASTER	0.017	8.7 MB 4 s	20.4 MB 2.9 s	10 MB 2.7 s	101 MB 2.3 h
LEATHER	0.014	17 MB 5.8 s	30 MB 3.3 s	23 MB 3.0 s	101 MB 3.2 h
PLASTIC	0.027	4.7 MB 3.7 s	18 MB 2.8 s	6.8 MB 2.3 s	56 MB 3.6 h
CUSHION	0.038	18.8 MB 9.3 s	27.2 MB 3.3 s	20 MB 3 s	35 MB 4.5 h

Table 4.2: Informal version of Table 3.1 based on the empirical experiments from Section 4.4. The computation and reconstruction times are given in h=hours, m=minutes or s=seconds respectively. According to Section 4.4.1 we selected the following reduced ranks in the SVD case: $r = 20$ for PLASTER and PLASTIC, $r = 40$ for LEATHER and $r = 80$ for CUSHION. For the other techniques parameters yielding a comparable RMSE were chosen.

only slowed down by a factor between 4 and 8 but the storage requirements are obviously increased by the same factor.

These results suggest that tensor factorization based on the Tucker decomposition can be useful only for off-line storage or if the whole tensor or at least large slices need to be reconstructed at once because in this case the reconstruction costs of the individual modes amortize. Anyway, it might be an interesting venue for future research to find tensor decompositions more appropriate for rendering.

4.5 Summary

In this chapter we analyzed the performance of SVD-based matrix factorization techniques for BTF compression. After presenting arrangements of the BTF measurement data as matrix or tensor we empirically compared stand-alone SVD against regular and adaptive block partioning and tensor factorization. As promised in the introduction of this chapter we present here the empirical version of Table 3.1. Table 4.2 compares practical fitting costs, storage requirements and reconstruction costs for a fixed RMSE per material. The chosen RMSE corresponds to a reasonable visual quality for standard SVD.

The conclusion we can draw from this chapter and particularly from Table 4.2 is that adaptive block-wise factorization seems to deliver be the best compromise between compression ratio and reconstruction cost. In any case, if runtime performance is the determining factor there is no other possibility than to use a block-wise factorization scheme. Block-wise factorization using regular blocks can be useful if fast fitting times are required. Other advantages of this technique would be more efficient filtering and an improved cache locality since data points adjacent in the BTF matrix remain adjacent in the compressed representation.

If it is about compression ratio then standard SVD will usually be the method of choice since it offers usually the best compression ratio for a given RMSE. Please note, that if only the compression ratio counts, additional expensive coding steps like in image compression might be applied to compress the data further but then the possibility of fast and random access reconstruction directly from the compressed representation will usually be lost.

In our experiments tensor factorization was able to beat standard SVD in terms of compression ratio only for one material (CUSHION) which essentially means that we could not confirm the results from [WWS+05]. This could be explained by the fact that Wang et al. used materials with a particulary high correlation along one or more of the tensor modes which was not the case for our test samples (except CUSHION). With no doubt tensor factorization is an interesting technique which is able to exploit correlations in high-dimensional datasets which cannot be exploited using only matrix factorization. Nevertheless, for BTFs and in particular for BTF rendering which requires efficient reconstruction the method is clearly not optimal. As our experiments have shown the correlation along the additional modes was not high enough to significantly reduce the RMSE compared to SVD for the same compression ratio. Even more severe in terms of the practical applicability of tensor factorization is the fact that the random access reconstruction of single BTF pixels which is required for rendering applications is unacceptable slow compared to matrix factorization techniques. Therefore, tensor factorization seems to be more suitable for applications requiring the decompression of whole sub-tensors at once like the visualization of time-varying multivariate volume data as demonstrated in [WXC+08].

CHAPTER 5

DATA-DRIVEN LOCAL COORDINATE SYSTEMS

In Chapter 4 we compared several BTF compression algorithms solely based on matrix factorization. These techniques do not exploit any prior knowledge about the geometry of the captured material. This seems a bit of a waste since it is a well known fact that the effectiveness of image-based representations is strongly related to the accuracy of the geometry the reflection data is parameterized on [GGSC96].
For example, in the extreme case, where the base geometry coincides with the actual geometry up to all visible details and correct local coordinate systems parameterizing the incoming and outgoing directions are given, the only information that needs to be stored is the "pure" light transport, i.e. the convolution of the surface BRDF (which subsumes the small, invisible surface details) with the incoming lighting plus inter-reflections. As shown by recent results the light transport matrix is of low rank in this case [MSRB07].

Unfortunately, in practice neither the geometry is known, because this would require an exact geometry reconstruction which is difficult to obtain from complex materials, nor the local coordinate systems are known because in case of an anisotropic BRDF this would require knowledge of the micro-geometry. The resulting effects are twofold:

1. **Parallax** - reflected radiance from physically different surface locations is projected onto the same location of the base geometry and thus treated within the same reflectance function ($\sim$same row in $\hat{\mathbf{B}}$).

2. **Misaligned BRDF** - the local coordinate system which parameterizes in- and outgoing directions is usually derived from the base geometry. Therefore, the parameterized BRDF measurement vectors (again: rows in $\hat{\mathbf{B}}$) from different surface points will be different even for objects with uniform BRDF (see Figure 5.1).

Both effects hamper the effectiveness of compression techniques based on statistics like co-variance because they introduce variance to the data matrices. The

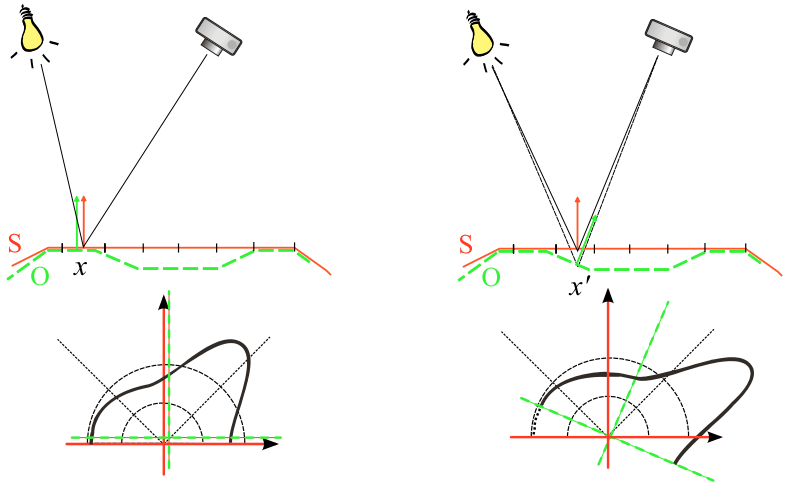


Figure 5.1: Small deviations of the base geometry $\mathbf{S}$ (red) from the real surface O (green) lead to misalignments in the reflectance data: the actually measured BRDF slice (bottom) is rotated and changes its shape depending on the surface slope. As a result the data at the points $\mathbf{x}$ and $\mathbf{x}'$ differs if put into the common coordinate system derived from $\mathbf{S}$ (red) even for identical surface BRDF.

factorization algorithms obviously can not decide in this case if the variance is part of the actual appearance or just resulting from one of these two effects.

Readers experienced in image-based rendering are of course familiar with these problems. In fact, these effects are inevitable in image-based rendering because exact geometry with screen-pixel accuracy and local frames aligned with the principal BRDF features are required to avoid them. Apart from the fact that such an accurate reconstruction of geometry and BRDF from images is difficult to acquire it also contradicts the original motivation of image-based rendering which was independence from scene complexity. This is especially true in the case of typical BTFs with their small and fuzzy geometry and the resulting complex light transport: the meso-scale geometry of BTFs is usually quite hard to reconstruct and an explicit representation within the rendering system would be very expensive in terms of storage and rendering. Therefore, in this chapter we propose an *intermediate* or *hybrid* representation residing between pure geometric and image-based representations. Loosely speaking, the method can be interpreted as an adaption of normal mapping to BTFs.

Originally, normal mapping [Bli78] was designed to simulate the shading variations resulting from small-scale geometry. The idea was to discard the small-scale geometry and to use only its normals in the shading computation. Thereby, normal mapping does not render correct masking, shadows and silhouettes but it

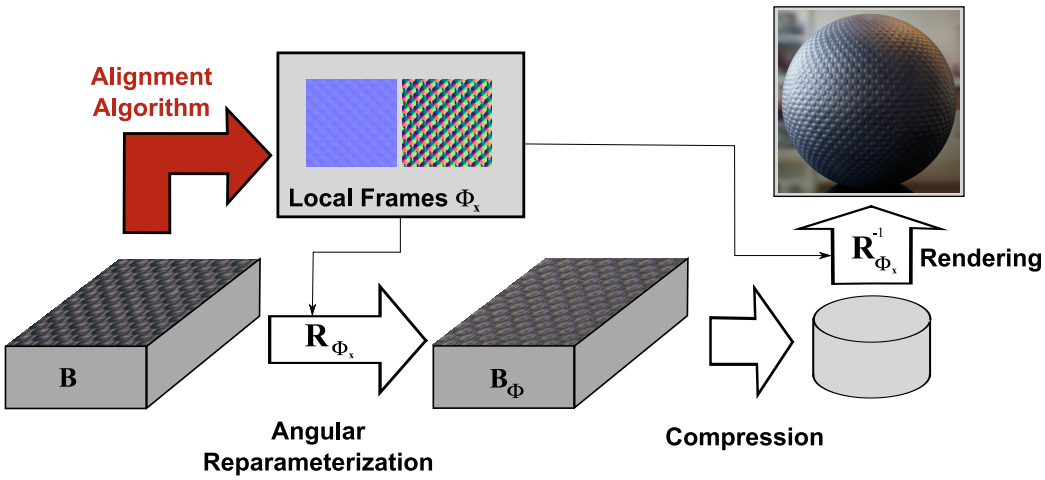


Figure 5.2: The principle behind our algorithm is the computation of consistent local frames (illustrated by the rgb-encoded normal and tangent maps) using a novel *alignment algorithm* and *angular reflectance reparameterization* according to these local frames. In the shown example (PLASTIC BTF) the data is then compressed using matrix factorization techniques. For rendering the original data is reconstructed by applying the inverse of the local frame rotation matrices.

creates a reasonable illusion of mesoscopic surface in the case of approximately flat surfaces.

We will exploit normal maps in a slightly different way: we perform an *angular reparameterization* of the measured reflectance data according to the map of local frames as illustrated in Figure 5.2. As mentioned above, we will not only employ normal maps for that purpose but full local frames given, e.g. as per-texel zyz-Euler angles $\Phi_{\mathbf{x}} = (\phi_{\mathbf{x}}, \theta_{\mathbf{x}}, \psi_{\mathbf{x}})$. Then the reparameterized BTF $\mathcal{B}_{V,\Phi}$ is given as follows:

$$\mathcal{B}_{V,\Phi}(\omega_i; \mathbf{x}; \omega_o) := \mathcal{B}_V(\mathbf{R}^T_{\Phi_{\mathbf{x}}} \cdot \omega_i; \mathbf{x}; \mathbf{R}^T_{\Phi_{\mathbf{x}}} \cdot \omega_o)$$

whereas

$$\mathbf{R}_{\Phi_{\mathbf{x}}} = \mathbf{R}_z(\phi_{\mathbf{x}}) \cdot \mathbf{R}_y(\theta_{\mathbf{x}}) \cdot \mathbf{R}_z(\psi_{\mathbf{x}})$$

is the per-texel rotation matrix that represents the local frame. The discrete version of $\mathcal{B}_{V,\Phi}$ can now be compressed using matrix factorization as exemplified in Chapter 4. We expect that the compression performance especially of correlation based algorithms like matrix factorization can be improved this way because variance resulting from deviations between real normals and normals from the reference surface (cf. again Figure 5.1) will be reduced even though parallax and masking effects are not corrected. During rendering, we simply need to apply the inverse rotation matrix $\mathbf{R}^{-1}_{\Phi_{\mathbf{x}}}$ to the in- and outgoing directions.

The question now immediately arises how these local frames will be obtained. A straight-forward solution could consist of a standard shape-from-shading or photometric-stereo method which estimates normals per discrete surface point. Unfortunately, such an approach would be quite limited: most photometric stereo methods like [RTG97] assume diffuse reflectance and they estimate a normal map for one viewpoint only. Furthermore, these approaches estimate only normals and not a full local frame which neglects the rotation around the normal which is important for anisotropic materials. Therefore, a novel algorithm for computing full local frames, i.e. normal *and* tangent vector, will be introduced in the following. We prefer to call this method an *alignment algorithm* because it tries to compute local frames which maximize the alignment between the per-texel reflectance functions.

In particular the approach consists of the following contributions:

- data-driven estimation of local coordinate systems without any assumptions about underlying reflectance properties, i.e. the method does not rely on a specific reflectance model
- improved compression performance of matrix factorization based compression algorithms with negligible additional run-time cost
- exploitation of the Spherical Harmonics Fourier-shift theorem which enables extremely fast computation of spherical correlations and might be of relevance for applications beyond compression like 3D texture reconstruction

The remainder of this chapter will be organized as follows. In Section 5.1 the core of our algorithm is developed for materials with nearly uniform BRDF. Then in Section 5.2 it is generalized to arbitrary materials. Results and experiments are the topic of the closing Section 5.3.

5.1 Local frame alignment for uniform materials

The key point of the algorithm is to cast the computation of local frames as a problem of *pairwise alignment* of reflectance functions. In order to highlight the differences between our method and standard photometric stereo let us first introduce a general photometric stereo framework.

In standard photometric stereo angular dependent image measurements $\mathbf{b}_{\mathbf{x}}(\omega_i, \omega_o)$ are compared against a BRDF model using a quadratic error functional per surface point $\mathbf{x}$:

$$E_{\mathbf{x}}(\alpha, \Phi) = \sum_{\omega_i \in I^d} \sum_{\omega_o \in O} |\mathbf{b}_{\mathbf{x}}(\omega_i, \omega_o) - f_{\mathbf{x}}(\alpha, \Phi; \omega_i, \omega_o)|^2 \tag{5.1}$$

Here α is a set of BRDF model parameters and I^d and O are the sets of measured view and light directions as in Chapter 4. Typically, only a normal direction (ϕ, θ) is estimated instead of a full local frame Φ, only a single viewpoint is used ($|O| = 1$) and a diffuse BRDF is assumed. In this case, the functional $E_{\mathbf{x}}$ can be minimized by solving a linear system of equations. In the case of general BRDF models the BRDF parameter set α has to be optimized, too, which usually requires a non-linear optimization which alternates between normal and BRDF parameter estimation (e.g. [GCHS05]).

The main problem of this standard approach regarding the improvement of BTF compression is the assumption of a specific BRDF model. This is inappropriate for materials with complex BRDFs or strong masking and light transport effects (inter-reflections, subsurface-scattering). One way to relieve this would be to allow a completely general reflectance function:

$$\check{E}_{\mathbf{x}}(\mathbf{a}, \Phi) = \sum_{\omega_i \in I^d} \sum_{\omega_o \in O} |\mathbf{b}_{\mathbf{x}}(\omega_i, \omega_o) - \mathbf{a}(\Phi; \omega_i, \omega_o)|^2 \tag{5.2}$$

with $\mathbf{a}(\Phi; \omega_i, \omega_o) := \mathbf{a}(\mathbf{R}^T(\Phi)\omega_i, \mathbf{R}^T(\Phi)\omega_o)$ where the rotation matrix $\mathbf{R}(\Phi) = R_z(\phi) \cdot R_y(\theta) \cdot R_z(\psi)$ parameterized by zyz-Euler angles $\Phi = (\phi, \theta, \psi)$ describes the local frame.

Of course, in this form the error functional does not make much sense since $\mathbf{a} = \mathbf{b}_{\mathbf{x}}$ and $\Phi = \mathbf{0}$ would be the trivial solution. Obviously, another constraint is required. While standard photometric stereo assumes a specific BRDF model our constraint is to assume a spatially uniform BRDF (we show in Section 5.2 how the functional can be generalized in order to deal with materials with non-uniform BRDFs, too). This results in the following error functional which integrates over the surface:

$$\mathcal{E}(\mathbf{a}, \Phi(\cdot)) = \sum_{\mathbf{x} \in E} \sum_{\omega_i \in I^d} \sum_{\omega_o \in O} |\mathbf{b}_{\mathbf{x}}(\omega_i, \omega_o) - \mathbf{a}(\Phi(\mathbf{x}); \omega_i, \omega_o)|^2 \tag{5.3}$$

This functional can be locally optimized using for example an EM-style algorithm (Expectation Maximization), which alternates between the estimation of a reflectance function $\mathbf{a}$ and the spatially varying map of local frames $\Phi(\mathbf{x})$ by minimizing the functional $\check{E}_{\mathbf{x}}(\mathbf{a}, \Phi(\mathbf{x}))$ while leaving $\mathbf{a}$ fixed (cf. Algorithm 1).

Unfortunately, this approach would be quite slow and prone to local minima. A global evaluation on a discrete grid of rotations $\{\Phi_i\}_{0<i<G}$ would also be infeasible. If we assume, for example, a desired accuracy of one degree along all three angular dimensions $\check{E}$ actually has to be evaluated for $360 \times 90 \times 360$ rotations.

Therefore, we propose to reformulate the objective function as a correlation:

$$\mathcal{C}(\mathbf{a}, \Phi(\cdot)) = \sum_{\mathbf{x} \in E} \sum_{\omega_i \in I^d} \sum_{\omega_o \in O} \mathbf{b}_{\mathbf{x}}(\omega_i, \omega_o)\mathbf{a}(\Phi(\mathbf{x}); \omega_i, \omega_o) \tag{5.4}$$

In order to understand what is won by this small change let us now assume that the reflectance functions are continuous and defined over the whole spheres of incoming and outgoing directions Ω_i and Ω_o. Then we can interpret the correlation between two continuous 4D reflectance functions $\mathbf{a}$ and $\mathbf{b}$ as a measure of alignment:

$$C_{\mathbf{x}}(\Phi) = \int_{\Omega_i} \int_{\Omega_o} \mathbf{b}_{\mathbf{x}}(\omega_i, \omega_o)\mathbf{a}(\Phi; \omega_i, \omega_o) d\omega_o d\omega_i. \tag{5.5}$$

If we assume that $\mathbf{a}$ and $\mathbf{b_x}$ are only rotated versions of each other then this correlation would reach its maximum for the aligning rotation. In other words, $C_{\mathbf{x}}(\Phi)$ represents the likelihood for the local frame Φ to align the rotated functions. Maximizing $C_{\mathbf{x}}(\Phi)$ now corresponds to the determination of the Euler angles $\bar{\Phi}$ which maximize the correlation between $\mathbf{b_x}$ and the rotated $\mathbf{a}$:

$$\bar{\Phi} = \arg\max_{\Phi} \; C_{\mathbf{x}}(\Phi) \tag{5.6}$$

By employing the Fourier Transform in the group of rotations $SO(3)$ this maximization can be computed in frequency space which is the key to the improved performance of our method. It is convenient for the derivation of this step to first reduce the complexity of the problem to the domain of 2D reflection functions defined over the sphere. A generalization to 4D reflectance functions will be given afterwards.

5.1.1 Spherical correlation

Modifying the above correlation term 5.5 to operate on functions defined on the sphere (and dropping the spatial index for convenience) leads to the following integral:

$$C^{\Omega}_{\mathbf{a},\mathbf{b}}(\Phi) \;\; = \;\; \int_{\Omega} \mathbf{b}(\omega)\mathbf{a}(\Phi; \omega) \; d\omega \tag{5.7}$$

This integral has the form of a spherical convolution such that the generalized Fourier shift theorem is applicable [KR03] which generalizes the basic Fourier shift theorem defined for 1D functions to functions defined over the sphere. Indeed, we can decompose bandlimited $\mathbf{a}$ and $\mathbf{b}$ with maximum bandwidth L into the Spherical Harmonics (SH) basis:

$$\begin{aligned} \mathbf{a}(\omega) &= \sum_{l=0}^{L} \sum_{|m|\leq l} \hat{a}^l_m Y^l_m(\omega) \\ \mathbf{b}(\omega) &= \sum_{l=0}^{L} \sum_{|m|\leq l} \hat{b}^l_m Y^l_m(\omega). \end{aligned}$$

The generalized Fourier shift theorem now stems from the fact that a rotated SH-basis function can be written as a linear combination of SH-basis functions of the same band:

$$Y_m^l(\mathbf{R}^T(\Phi)\omega) \;=\; \sum_{|k|\leq l} D_{km}^l(\Phi) Y_k^l(\omega) \tag{5.8}$$

Here $D_{km}^l(\Phi)$ denotes a Wigner-D function. From (5.8) follows

$$\mathbf{b}(\mathbf{R}^T(\Phi)\omega) = \sum_{l=0}^{L} \sum_{|k|\leq l} \left(\sum_{|m|\leq l} \hat{b}_m^l D_{km}^l(\Phi) \right) Y_k^l(\omega) \tag{5.9}$$

which by exploiting the orthogonality of the SH basis immediately leads to

$$C_{\mathbf{a},\mathbf{b}}^{\Omega}(\Phi) = \sum_{l=0}^{L} \sum_{|k|\leq l} \sum_{|m|\leq l} \hat{a}_k^l \overline{\hat{b}_m^l D_{km}^l(\Phi)} \tag{5.10}$$

Since the right hand side of Equation (5.10) has the form of the Fourier decomposition of $C_{\mathbf{a},\mathbf{b}}^{\Omega}(\Phi)$ in the rotation group $SO(3)$ the generalized Fourier shift theorem follows immediately by comparing coefficients:

$$\hat{c}_{mn}^l = \hat{a}_m^l \overline{\hat{b}_n^l} \tag{5.11}$$

which now (analogously to convolutions in the planar domain) allows to compute the Fourier coefficients $\hat{c}_{mn}^l$ of $C_{\mathbf{a},\mathbf{b}}^{\Omega}(\Phi)$ by multiplying the Fourier coefficients of $\mathbf{a}(\omega)$ and $\mathbf{b}(\omega)$ (in fact, an outer product has to be computed per band).

The direct evaluation of Equation (5.10) has complexity $\mathcal{O}(L^3)$. Fortunately, there exists an (inverse) Fast Fourier Transform (FFT) [KR03] which evaluates Equation (5.10) for a grid of $(2L+1)^3$ rotations with complexity $\mathcal{O}(L^3 \log^2 L)$ compared to $\mathcal{O}(L^6)$ for the direct evaluation.

We implemented both the direct evaluation of Equation (5.10) and its efficient evaluation by applying inverse FFT using the freely available C-library SOFT [SOF]. Furthermore, we implemented the "brute-force" evaluation of Equation (5.7) using numerical integration. Table 5.1 shows timing results for different values of L.

5.1.2 Generalization to 4D

To generalize the results from the previous section to the correlation of 4D reflectance functions in Equation (5.5) we follow a tensor product approach, i.e. we

L	Eqn. (5.7) $\lvert\tilde{\Omega}^+\rvert = 151$	Eqn. (5.10) direct	Eqn. (5.10) FFT
4	0.325	0.685	0.029
8	1.873	27.673	0.336
16	15.144	>1000	5.326

Table 5.1: Average computation times (in milliseconds) for spherical correlation on an $\mathcal{O}(L^3)$ grid for varying L on an AMD Athlon 3200+.

represent reflectance functions in the tensor product basis of Spherical Harmonics:

$$\mathbf{a}(\omega_v, \omega_l) = \sum_{l_1,l_2}^{L} \sum_{m_1,m_2} \hat{a}_{m_1 m_2}^{l_1 l_2} Y_{m_2}^{l_2}(\omega_v) Y_{m_1}^{l_1}(\omega_l)$$

Repeating the steps from the previous subsection (we omit them here for brevity) leads to the following lengthy sum

$$C_{\mathbf{a},\mathbf{b}}^{\Omega\times\Omega}(\Phi) = \sum_{l_1 l_2}^{L} \sum_{m_1 m_1'} \sum_{m_2 m_2'} \hat{a}_{m_1 m_2}^{l_1 l_2} \overline{\hat{b}_{m_1' m_2'}^{l_1 l_2} D_{m_2 m_2'}^{l_2}(\Phi) D_{m_1 m_1'}^{l_1}(\Phi)} \tag{5.12}$$

After a simple reordering of terms the direct evaluation of this sum has complexity $\mathcal{O}(L^5)$ which would leave us with an expensive $\mathcal{O}(L^8)$ algorithm. Applying the inverse FFT in a tensor product fashion will result in a still expensive $\mathcal{O}(L^6 log^2 L)$ algorithm and returns the correlation for arbitrary combinations of rotations $(\Phi_1, \Phi_2) \in SO(3) \times SO(3)$ while we require only those elements where $\Phi_1 = \Phi_2$. We currently have not found a way to exploit this.

Another problem of the tensor product representation is its memory requirement. For $L = 8$ each reflectance functions consists of 4096 SH coefficients which easily sums up to hundreds of megabytes of data for typical datasets. Therefore, we propose to factorize the reflectance functions $\mathbf{a}, \mathbf{b}$ into a sum of products of 2D functions which reduces both the memory requirements and computational complexity. Specifically, we find representations of the form

$$\mathbf{a}(\omega_v, \omega_l) \approx \sum_{i}^{K} a_{1,i}(\omega_v) a_{2,i}(\omega_l)$$

$$\mathbf{b}(\omega_v, \omega_l) \approx \sum_{i}^{K} b_{1,i}(\omega_v) b_{2,i}(\omega_l)$$

using for example SVD. Substituting these representations for $\mathbf{a}$ and $\mathbf{b}$ leads to a modified correlation term

$$\tilde{C}_{\mathbf{a},\mathbf{b}}^{\Omega\times\Omega}(\Phi) = \sum_{i}^{K} \sum_{j}^{K} C_{a_{1,i},b_{1,j}}^{\Omega}(\Phi) C_{a_{2,i},b_{2,j}}^{\Omega}(\Phi) \tag{5.13}$$

L	Eqn. (5.5)	Eqn. (5.12)	Eqn. (5.13) K=4	K=6
4	291.2	26.86	1.145	2.591
8	>1500	>3500	10.683	23.803
16	N.N.	N.N.	195.937	445.118

Table 5.2: Average computation times (in milliseconds) for computing the correlation of 4D-reflectance functions on an $\mathcal{O}(L^3)$ grid for varying L on an AMD Athlon 3200+. Direct approaches are not feasible for $L > 4$.

which reduces the 4D correlation to a sum of products of 2D correlations. Using again the inverse FFT for spherical correlations we now have an $\mathcal{O}(K^2 L^3 \log^2 L)$ algorithm. In our experiments we set $4 \leq K \leq 8$ but typically 4 components suffice without degrading the results since the main features relevant for the alignment are captured in this representation.

As in the 2D case we implemented the direct evaluation of Equation (5.12) and simple numerical integration for comparison. The huge performance benefits gained from Equation (5.13) are illustrated in Table 5.2. Using the settings $L = 8$ and $K = 4$ we align about one hundred reflectance functions per second for a sufficiently dense grid of rotations.

5.1.3 Putting it Together

Now the ingredients required to formulate the maximization algorithm for functional $\mathcal{C}(\mathbf{a}, \Phi(\cdot))$ can be put together. As already indicated the algorithm will alternate between the estimation of an alignment function $\mathbf{a}$ and the estimation of the local coordinate frames $\Phi(\mathbf{x})$ which employs the correlation computation in frequency space.

Given a set of reflectance functions $\{\mathbf{b}_\mathbf{x}\}_{\mathbf{x}\in E}$ we simply compute $\mathbf{a}$ as their mean. Then, iteratively, each $\mathbf{b}_\mathbf{x}$ is aligned to $\mathbf{a}$ using the factorized correlation equation (5.13). Since usually $L = 8$ the discrete grid of rotations is still relatively sparse. Therefore, the strategy will be to use the location of the found maximum as initialization for an additional non-linear optimization step using, e.g. the Levenberg-Marquardt algorithm. These steps are summarized in Algorithm 1

Given the computation times for a single alignment the computation time for a whole BTF with an extent of 256^2 texels adds up about to about one hour using four iterations in Algorithm 1 and including the preprocessing time required for factorization, SH projection and non-linear optimization.

Algorithm 1 Alignment of reflectance functions $\{\mathbf{b_x}\}_{\mathbf{x}\in E}$

$j = 0$
$N = |E|$
$\mathbf{a}_0 = \mathbf{b}_{\mathbf{x}_0}$ {choose $\mathbf{x}_0 \in E$ randomly}
repeat
  for $\mathbf{x} \in E$ **do**
    $\Phi_{\mathbf{x}} = \text{argmax}_\Phi \; C_{\mathbf{a}_j,\mathbf{b_x}}(\Phi)$
  end for
  $\mathbf{a}_{j+1}(\omega_i, \omega_o) = \frac{1}{N} \sum_{\mathbf{x}\in E} \mathbf{b_x}(\Phi_{\mathbf{x}}; \omega_i, \omega_o)$
  $j = j + 1$
until $\epsilon < \tau$ {ϵ is the increase in accumulated correlation $\sum_{\mathbf{x}\in E} C_{\mathbf{a}_j,\mathbf{b_x}}(\Phi_{\mathbf{x}})$ between two iterations}

return Set of local frames $\{\Phi_{\mathbf{x}}\}$

5.2 Local-frame alignment for materials with non-uniform BRDFs

Most objects are made of different materials which may vary across the surface. Applying the approach from Section 5.1 will then align reflectance functions which have been measured from probably very different BRDFs which may lead to at least sub-optimal results. Our solution is to apply a k-means clustering algorithm with an orientation invariant distance metric. The metric is based on a normalized version of the correlation term (5.5):

$$d(\mathbf{a}, \mathbf{b}) = 1 - \max_{\Phi} \bar{C}_{\mathbf{a},\mathbf{b}}(\Phi) \tag{5.14}$$

$d(\mathbf{a}, \mathbf{b})$ defines a so-called semi-metric (triangle inequality does not hold) and we use it as an orientation invariant distance metric for the k-means clustering. By using this metric a reflectance function will be assigned to the cluster where the correlation between the cluster center and the suitable rotated reflectance function is maximized.

5.3 Experiments and Results

To test our algorithm we applied it to synthetic data, measured BTFs and a far-field surface reflectance field, i.e. a BTF parameterized on a 3D-object instead of a simple plane. To evaluate the improvement in compression performance for the measured data global matrix factorization (see Chapter 4) was applied to the datasets before and after alignment.

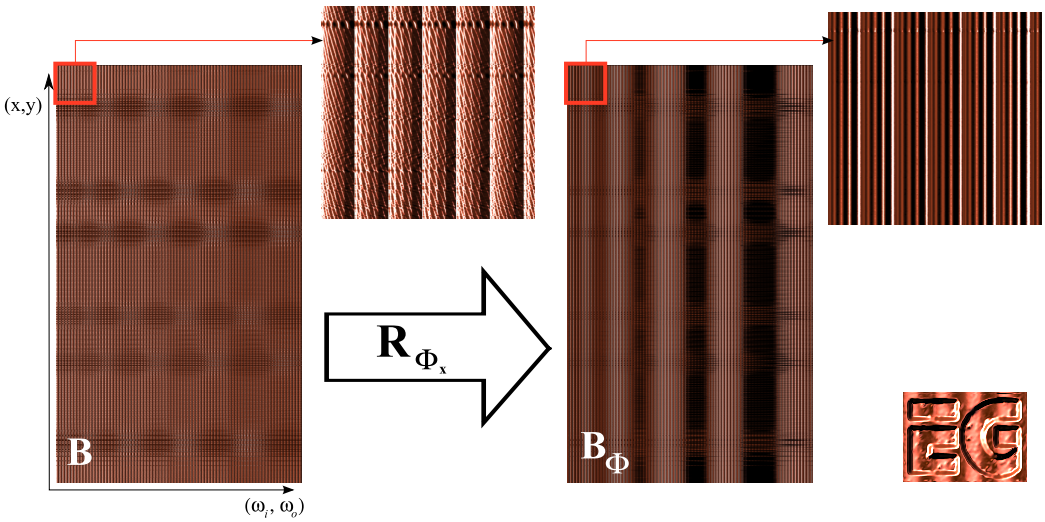


Figure 5.3: Synthetic test: the original data matrix $\hat{\mathbf{B}}$ of the material shown in the lower right is turned into an aligned data matrix $\hat{\mathbf{B}}_{\Phi_x}$ by our algorithm. The zoom-ins into the matrix images demonstrate how well the angular features are aligned – the diagonal features have been turned into almost perfectly vertical features.

5.3.1 Synthetic Data

The basic principle of the algorithm was tested by creating a synthetic dataset from a heightfield ("noisy" Eurographics logo), a tangent map (modulated by a cosine function), and a single anisotropic BRDF (Ashikhmin-Shirley BRDF model [AS00]) without computing interreflections. Since this dataset has no registration errors and measurement noise, variance in the data matrix results only from parallax and wrong local coordinate systems. We sampled this synthetic BTF at $81 \times 81 = 6561$ directions and applied the alignment algorithm described in the previous section. Using three iterations and the settings $L = 8$ and $K = 5$ the alignment took about five minutes. Figure 5.3 shows the results. Note how well the angular features of the reflectance functions (rows of $\hat{\mathbf{B}}$) are aligned resulting in a data matrix with almost identical rows. Obviously, such a matrix has a significantly lower rank than the original matrix.

5.3.2 Measured BTFs

In order to test the algorithm also with real world data containing registration errors, measurement noise and significant masking it was applied to the measured BTFs from Figure 4.1. The results are shown in Figures 5.4 and 5.5. The plots show that the reduction in RMSE is quite significant compared to standard SVD.

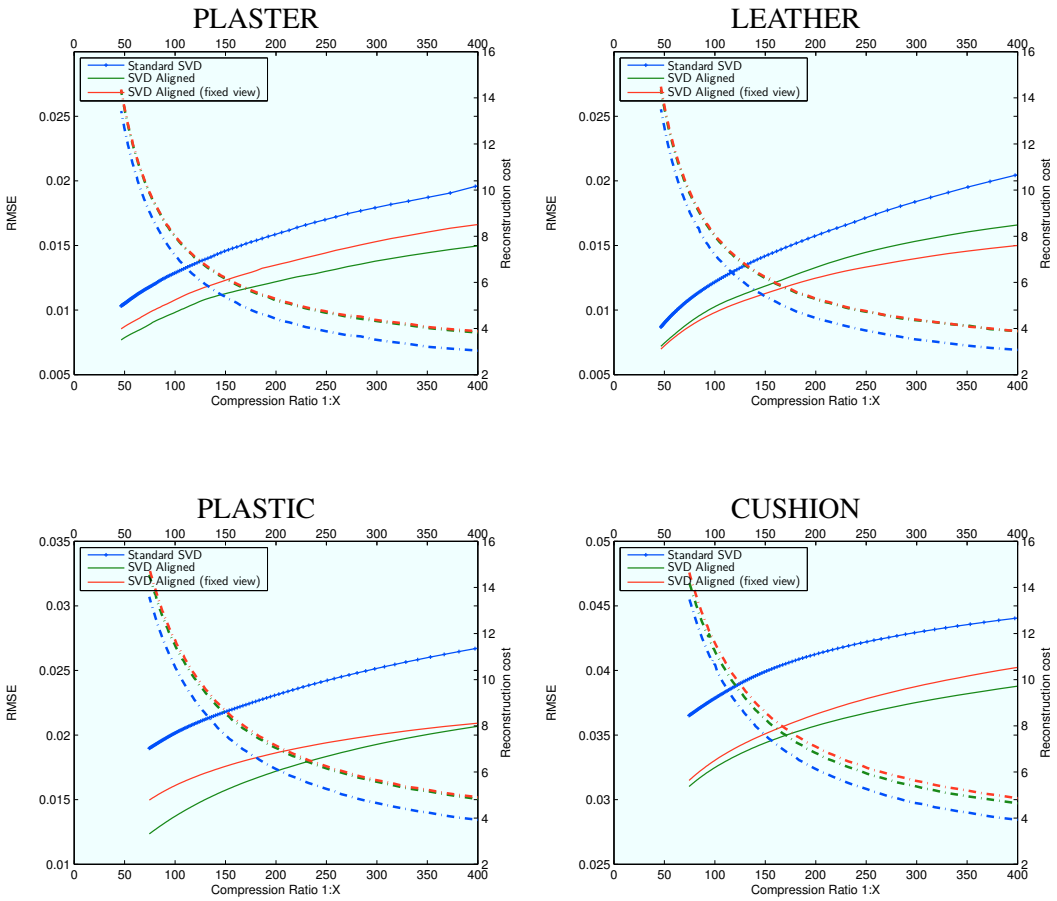


Figure 5.4: RMSE and reconstruction cost of global factorization (standard SVD) for original data and after reparameterization to local frames. For all materials tested, the RMSE is reduced for the same compression ratio after applying the alignment. Usually, computing the alignment between full 4D reflectance functions using the factorized approximation performance best. The additional reconstruction cost from local frame rotations is almost negligible.

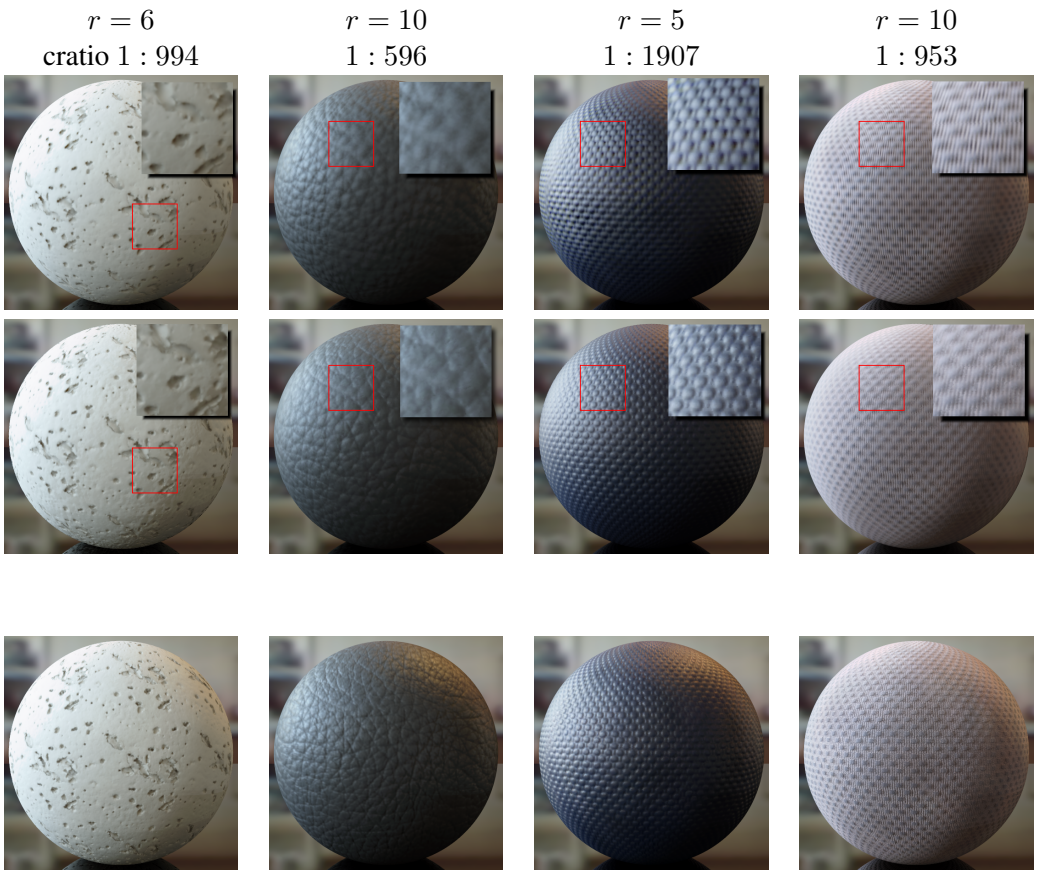


Figure 5.5: Visual comparison of global factorization (rank-r approximation) without (first row) and with (second row) angular reparameterization using the local frames computed by our alignment algorithm. Spatial and angular blur is reduced significantly especially for the high compression ratios shown. The bottom row shows reference images.

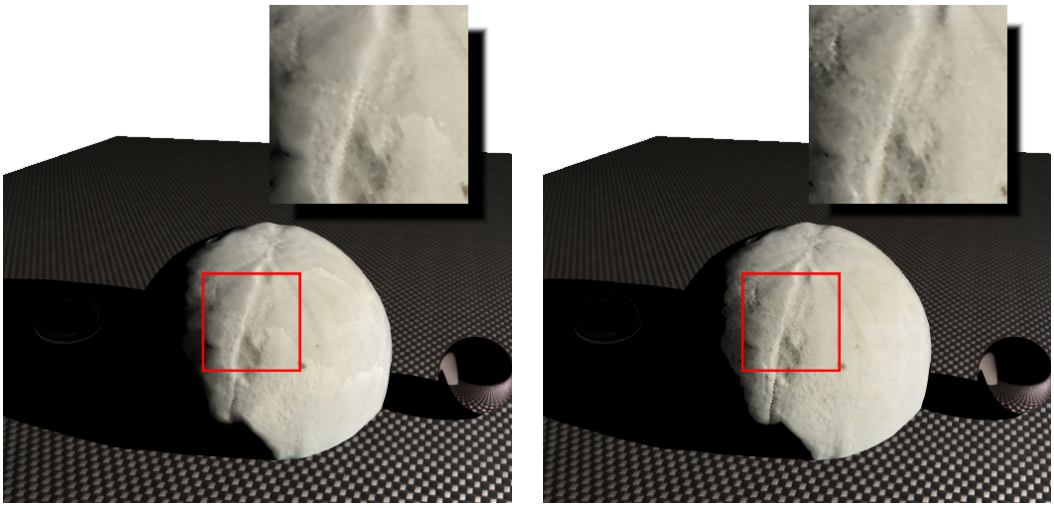


Figure 5.6: Compressing BTF data measured on 3D geometry, in this case a fossil echinite. The left image shows the result using adaptive block-wise factorization ($p_n = 32$, $p_m = 1$, $r_b = 8$) without alignment of local coordinate systems. On the right the result after alignment with our method is shown.

In all cases the alignment of the full 4D reflectance functions out-performs the use of only a single fixed view and reconstruction cost increases only slightly. The improved RMSE results also in an improved visual quality especially for low-rank approximations as shown in Figure 5.5. Nevertheless, the effect is not as impressive as the plots suggest. One reason for this might be the additional resampling step involved in applying the local coordinate system transformation before factorization.

5.3.3 Measured Far-Field Surface Reflectance Field

An interesting application of our method is the compression of BTF data parameterized on a 3D surface instead of a plane. In this case the measured reflectance data is usually parameterized by the local coordinate systems of the mesh triangles. Since the geometry of the proxy geometry mesh often deviates significantly from the real geometry, the deviations between the real local coordinate systems and the ones derived from the base geometry are usually higher than in the case of a planar material sample. As sketched in the introduction of this chapter this means increased variance or equivalently increased rank of the light transport matrix.

To demonstrate the effectiveness of our method in this particular case we took the BTF data of a fossil echinite which was presented in our paper about rapid

acquisition of geometry and BTF data [MBK05] and estimated per-texel local coordinate systems for it using the alignment algorithm before compressing the data using adaptive block-wise factorization. As shown in Figure 5.6 the quality of the approximation is significantly improved, e.g. the borders between different blocks have become almost invisible and the small-scale structures and shadows are much sharper.

5.4 Summary

In this chapter a method which estimates local coordinate systems from image-based measurements was presented. In contrast to comparable techniques from the field of photo-metric stereo the algorithm does not require assumptions about an underlying reflectance model but operates directly on the measurements. The local coordinate systems are computed by finding rotations which maximize the correlation between the reflectance functions of the data matrix. This way the variance present in the data matrix is reduced which leads to an improved performance of factorization algorithms based on SVD.

The algorithm was made efficient by projecting the reflectance functions onto the Spherical Harmonics basis and evaluating the correlation term in Fourier space. Then the Fast Fourier Transformation defined for functions on the rotation group, $SO(3)$, can be applied which enables a dramatic improvement in run-time performance. The complexity for the evaluation of the correlation is reduced from $\mathcal{O}(L^8)$ for the direct sum to $O(K^2L^3\log^2 L)$ by using FFT.

In the examples presented in Section 5.3 the method was applied to synthetic data and measured BTF. The results demonstrate that the data alignment achievable by the algorithm leads to significant variance reduction which improves the compression performance of matrix factorization algorithms.

CHAPTER 6

RELATED AND CONCURRENT WORK

We will close the compression part of this thesis by briefly reviewing existing BTF compression and modeling techniques not based on matrix factorization. Furthermore, we will introduce general photometric stereo algorithms and other ideas for the alignment of reflectance functions since they inspired our alignment algorithm for computing data-driven local coordinate systems.

6.1 Fitting Parametric Models

As mentioned above the rows $\hat{\mathbf{b}}^{\mathbf{x}}$ of the discrete BTF matrix $\hat{\mathbf{B}}$ are per-texel 4D reflectance functions which resemble BRDFs. Therefore, it sounds reasonable to fit the parameters of an analytical BRDF model $f(\mathbf{p}_{\mathbf{x}}; \omega_i, \omega_o)$ for each row of $\hat{\mathbf{B}}$. *Fitting* in this case means finding a set of parameters minimizing a given error criterion:

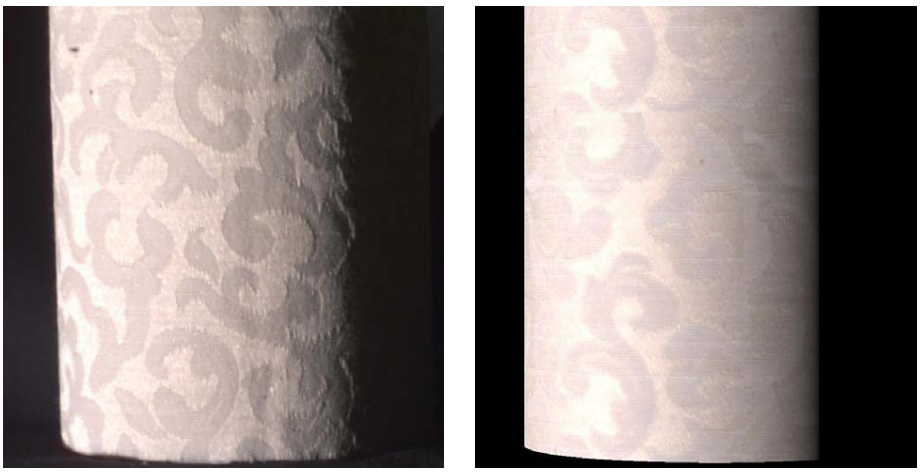
$$\mathbf{p}_{\mathbf{x}}^{min} = arg \min_{\mathbf{p}_x} \mathcal{E}\left(\hat{\mathbf{b}}^{\mathbf{x}}, f(\mathbf{p}_x; \cdot, \cdot)\right)$$

A typical error metric is the Euclidian distance. The resulting spatially varying coefficient map $\mathbf{p}_{\mathbf{x}}^{min}$ can then be further compressed using an image compression algorithm.

This approach was introduced by McAllister et al. [MLH02] and they used the Lafortune model [LFTG97] (Equation 2.7). Other models like the Cook-Torrance model (Equation 2.6) have also been used in the literature (e.g. [GTR$^+$06]).

While this approach is elegant and allows high compression ratios as well as real-time reconstruction it turns out that using analytical BRDF models for the representation of reflection functions does not work for materials with a certain depth. In this case the influence from neighboring geometry grows stronger and violates the assumptions of analytical BRDF models like reciprocity or statistically independent distribution of micro-facets. This results in a loss of depth impression in the rendered images as depicted in Figure 6.1. Therefore, representations have been developed that try to take theses influences into account by using more complex analytical representations of the reflectance functions.

In [MMK04a] it was proposed to interpret each sampled reflectance function



(a) (b)

Figure 6.1: (a) Photograph of a real cylinder covered with white fabric. (b) Rendered cylinder using a Lafortune-model SVBRDF fitted to measured BTF of white fabric. Note the loss of depth impression (images from [McA02]).

$\hat{\mathbf{b}}^{\mathbf{x}}$ as a set of per-view 2D reflection functions $b^{\mathbf{x}}_{\omega_o}$ that vary with lighting direction. Then each 2D reflection function is approximated independently using a parametric model like biquadratic polynomials or cosine lobes. This approach significantly improves the approximation quality compared to BRDF-models but the fidelity possible with techniques based on matrix factorization is not reached because by using analytical functions with a restricted shape visually important features in the data might be missed. Furthermore, the memory requirements are increased by a factor equal to the number of fixed view directions used.

A variant of this approach requiring even more memory by fitting an additional non-linear mapping per 2D reflection function was proposed in [FH04].

6.2 Fixed Basis Projection

Fixed function bases like the Fourier basis used in the Discrete Cosine Transform (DCT) or Wavelets are quite successful in data- and especially in image-compression. This success can partly be explained by the fact that an adapted data-driven basis usually does not amortize in the case of image compression because the space required to store the basis itself eliminates the improvement gained from the reduced number of coefficients. For huge image-based data sets consisting of thousands of images this relation changes towards the data-driven basis. As a result only few literature exists which explicitly reports about the application of fixed function basis transformation to BTF and relightable image data.

One example is in [WL03] Wong et al. who inspected the per-texel data $\hat{\mathbf{b}}^{\mathbf{x}}$ and projected it onto the SH basis. Since the SH basis is defined on the sphere and the reflectance functions are 4-dimensional the SH-transform has to be applied to the lighting and viewing dimensions consecutively yielding the following approximation:

$$\hat{\mathbf{b}}^{\mathbf{x}}(\omega_i, \omega_o) \approx \sum_{j}^{m} \sum_{k}^{n} b_{jk} Y_j(\omega_i) Y_k(\omega_o) \tag{6.1}$$

Here the $Y_j(\omega)$ are the SH-basis functions and the coefficient matrix b_{jk} is computed using double projection:

$$b_{jk} = \int_{\Omega} \left(\int_{\Omega} \hat{\mathbf{b}}^{\mathbf{x}}(\omega_i, \omega_o) Y_k(\omega_o) d\omega_o \right) Y_j(\omega_i) d\omega_i.$$

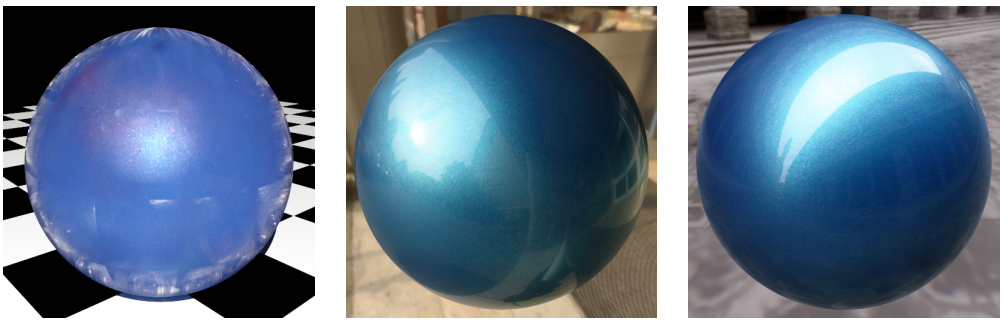
Using this representation, typically compression ratios around 1:10 can be achieved without loosing too much visual fidelity. In order to improve upon this situation it was proposed in [HWL05] to compress the coefficients using PCA. A drawback of the approach is the increased run-time cost since the SH-reconstruction sum (Equation (6.1)) has to be evaluated. Furthermore, high-frequency content like specular peaks will be smoothed out if higher order coefficients are discarded which makes SH-based compression suitable only for moderately specular materials.

Generally, most of the fixed function basis transformation techniques can be combined with matrix factorization, as it has been also demonstrated by Ma et al. [MCT+05] who transformed BTF data using the Laplacian pyramid before the application of PCA. The paper shows also that hierarchical function bases can be helpful for representing multi-scale data. A thorough overview on the application of several fixed function bases (Polynomials, Spherical Harmonics, Wavelets) for the compression and smooth reconstruction of 2D-reflectance functions in the context of image-based relighting is presented in [MPDW04].

6.3 Hybrid Models

Matrix factorization techniques work out well if the covariance matrix has low rank. For materials whose appearance is dominated by high-frequency stochastic patterns like pearlescent metallic paints which contain a large number of randomly distributed and oriented mirror-like effect flakes (cf. Figure 1.1) this is not the case.

Therefore, Rump et al. [RMS+08] proposed a hybrid BTF model which consists of the following two parts:



(a) (b) (c)

Figure 6.2: (a) Rendering of metallic car-paint BTF using standard BTF compression. Several artifacts like tiling and blurred specular reflection destroy the realistic impression. (b)+(c) The hybrid model renders artifact-free images (images from [RMS+08]).

- The homogenous BRDF part, which describes the reflection behavior of the base and top layer of the paint and is represented by a multi-lobe Cook-Torrance BRDF model (Equation 2.6). Since modern effect paints can exhibit excessive angular color shifts the model is extended by an angular dependent color table.

- The spatially varying part, which models the effect particles and consists of small BTF patches from which the homogenous part has been subtracted. During run-time these patches are pasted using a texture synthesis algorithm to create an arbitrarily large texture without visible repetition artifacts.

This way high-frequency content in both spatial *and* angular dimensions can be preserved because two different representations are used to model them. For paints this approach is feasible because in this case the depth variation on the sample can be neglected which allows a robust fitting of both parts to BTF measurements as demonstrated in Figure 6.2.

6.4 MRF-Modeling

An important issue when it comes to the practical application of BTFs is *texture synthesis*. Since the measured samples usually have a quite limited spatial extent and resolution (usually around 256^2 texels) a method is required which enlarges these samples without visible repetition artifacts in order to wrap them seamlessly onto real geometry. Using such synthesis techniques also for editing of BTFs will be the main topic of Chapter 8.

Since the principle of texture synthesis is to generate a larger texture from a small sample it can also be interpreted as a data compression technique. If a *sample-based* texture synthesis technique in the spirit of Efros and Leung [EL99] is used, which copies appropriate samples from the original texture, this compression is realized of course only in relation to the synthesized texture.

While these non-parametric texture synthesis techniques currently enjoy great popularity there are also parametric approaches around which try to represent textures using statistical models.

Pioneering work in this area was performed in the group of Michal Haindl at UTIA, Prague. In [HF03] they presented a BTF modeling technique based on Gaussian Markov random fields (GMRF). In follow-up papers 3D causal autoregressive models (CAR) [HFA04] and per-subspace 2D CAR [HF07] were proposed.

Since these models usually have difficulties in maintaining low-frequency spatial structures they have to be combined with an approximate 3D reconstruction of the sample which can obtained, e.g. using photometric stereo. During rendering this depth map is used to simulate shadowing and masking effects while the color variation is synthesized from the statistical model.

While modeling BTFs using parametric statistical models is a promising direction for future research the approximation quality of currently available models does not suffice for high quality rendering.

6.5 Photometric Stereo and Alignment of Reflectance

Our work from Chapter 5 is actually the first that computes local coordinate frames in order to improve the performance of matrix factorization for compression of image-based data matrices. The only method that also computes local frames not only for the purpose of 3D reconstruction but for improving the representation of measured materials is the work of Lensch et al. [LKG+03]. They integrated local frames in the fitting process of spatially varying Lafortune BRDF parameters. Since they assume isotropic materials they only need to fit a normal vector per surface point.

Like this method, our algorithm is also closely related to work from the field of photometric stereo for general, non-lambertian BRDFs like [GCHS05] which actually can be interpreted as generalization of [LKG+03]. As sketched in Section 5.1 these techniques usually define a quadratic error functional and optimize for normals and BRDF model parameters. In contrast, our method maximizes a correlation term in order to determine local frames and computes representative sampled reflectance functions in a fully data-driven way. Thereby, our focus lies on improving data compression and not on an accurate 3D reconstruction. Never-

theless, it will be interesting venue for future research if our technique can also be used for 3D reconstruction.

The idea of aligning reflectance function in order to improve the compression of image-based rendering data was also pursued by Wood et al. [WAA+00]. They proposed to reflect surface light field data around the normal in order to increase the alignment of the reflective peak. Computing a custom alignment based on the correlation of reflectance functions as we do can be interpreted as generalization if this basic technique.

Part II

Editing

CHAPTER 7

BACKGROUND

A main deficiency of complex image-based material representations like the BTF is the lack of effective editing methods. As a consequence the use of BTFs in digital content creation applications like movies or advertisement is currently limited because these applications require advanced editing capabilities in order to realize the art director's concept.

Currently, there are in principle two different approaches for editing material appearance in Computer Graphics. The first approach is based on analytical and procedural models for reflectance and texture. Here the appearance is edited by changing the input parameters of the models. The second technique uses photo-editing techniques like filtering, cloning, painting etc. to manipulate the texel values of texture maps directly.

Unfortunately, both approaches cannot be easily generalized to complex representations like the BTF. A model-based approach contradicts the original motivation for image-based materials: using BTFs in cases where reflectance models are visually inferior. Photo-editing a BTF using standard image-editing software tools is difficult because each editing operation must be consistently propagated to the whole dataset.

However, since powerful BTF-models with an intuitive set of parameters still seem to be out of reach the second approach seems to be more amendable to generalization. In order to apply standard image editing operations to high-dimensional data we state that two additional components are required. First, we need a low-dimensional representation of the feature we want to edit. Preferably, this would be a single image or a set of curves or sliders. We will call this low-dimensional representation the *feature-of-interest model*. As suggested by its name this model does not need to represent all visually relevant features of the material which greatly simplifies the modeling step compared to a full material model. Then we need a *back-propagation model* which describes how the editing operations which have been applied to the feature-of-interest model are propagated to the full image-based representation. Again, this model can be much simpler then a complete material model because it can operate on the data.

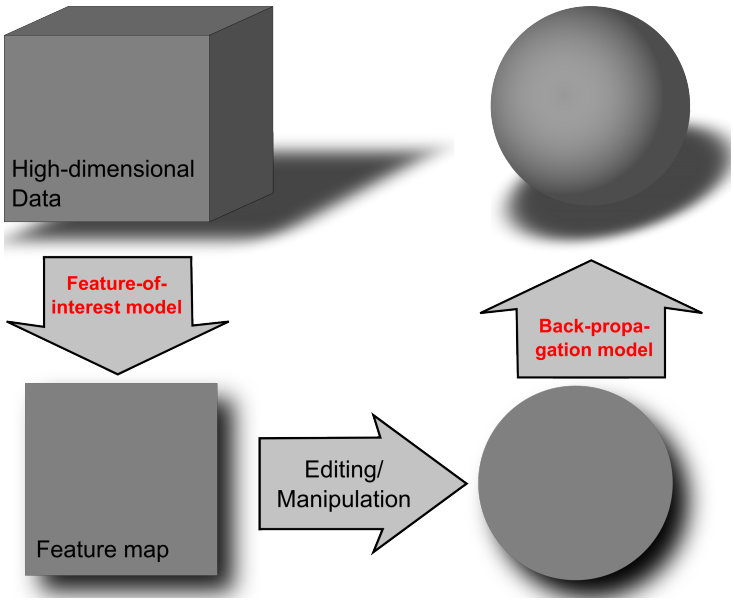


Figure 7.1: The back-propagation based editing scheme for high-dimensional data. The user has to deal only with a low-dimensional feature-of-interest map which is derived using a feature-of-interest model. Then the applied edit operations have to be propagated back to the whole data using the back-propagation model.

We will call such an editing scheme the *back-propagation based editing scheme*. It is illustrated in Figure 7.1 and consists of two main components:

1. **Feature-of-interest model** An editable representation like a set of sliders, curves or images which allows to edit a specific feature of interest. This representation should be computable from the original data in a fast and fully automatic way.

2. **Back-propagation model** The edit operations applied to the feature map have to be consistently propagated to the whole data. Usually new data is created which reflects the feature-specific editing operations.

A classical example of this editing scheme are image-warping methods where a 2D image is edited by manipulating a set of control points which usually correspond to important features in the image. Using an interpolation technique like thin-plate splines the editing operations applied to single control points are propagated to the whole image.

Please note the difference to dimensionality reduction techniques which also are candidates for data-driven editing like demonstrated by Matusik et al. [MPBM03]

or Lawrence et al. [LBAD+06]. Both approaches are similar in spirit to model-based editing techniques because the - in this case data-driven - model is manipulated (more or less) directly. Therefore, the visual quality that can be achieved also depends on the capabilities of the model. Unfortunately, Lawrence et al's Inverse Shade Trees are only practical for SVBRDFs which can be faithfully represented by only a few factors and it does not work for BTFs with meso-structure. The data-driven reflectance model from Matusik et al. requires a low-dimensional representation of a whole database of measured materials which then allows meaningful interpolation of the measured materials. Such a representation has not been found for BTFs yet.

The BTF editing technique we propose in this thesis consists of two different components. The first component, which will be presented in chapter 8, is based on a back-propagation based editing scheme and will be used to edit spatially varying features of the BTF. We solve the back-propagation problem by adapting the Image Analogies framework [HJO+01] for BTFs – *BTF Analogies*.

To edit also the micro-scale reflectance in the second component of our technique we propose an interpolation-based approach resembling the texture database editing of Matusik et al. [MZD05]. It consists of interpolating between different BTFs but only *after* they have been made "compatible" using our BTF Analogies technique – *analogy-driven* BTF interpolation. An interesting variant of the technique consists of a combination of the analogy-driven interpolation with a procedural model for meso-structure to create a combined procedural and data-driven model for leather BTFs which for the first time offers a fully parameterized BTF model for a particular material class and allows for smooth navigation within the space of the measured samples.

Because the adaption of texture transfer and the Image Analogies framework is central to our method we will now give a short introduction into these topics.

7.1 Texture Transfer and Image Analogies

Texture transfer and *constraint* texture synthesis were originally developed to introduce a certain degree of user control into texture synthesis (e.g. [Ash01]). While pure texture synthesis aims at generating an arbitrary large version of a small exemplar texture without visible repetition artifacts (Figure 7.2, left) texture transfer gives control over the synthesis process by defining an image which is used to constraint the distribution of textural features. This works fine for simple RGB-textures (Figure 7.2, middle and right) but is impractical for high-dimensional textures like BTFs because the constraint has to be compatible with the source data in order to guide the search for best-fitting pixels/patches. Essentially, this means that the transfer target has to be a BTF, too, which would

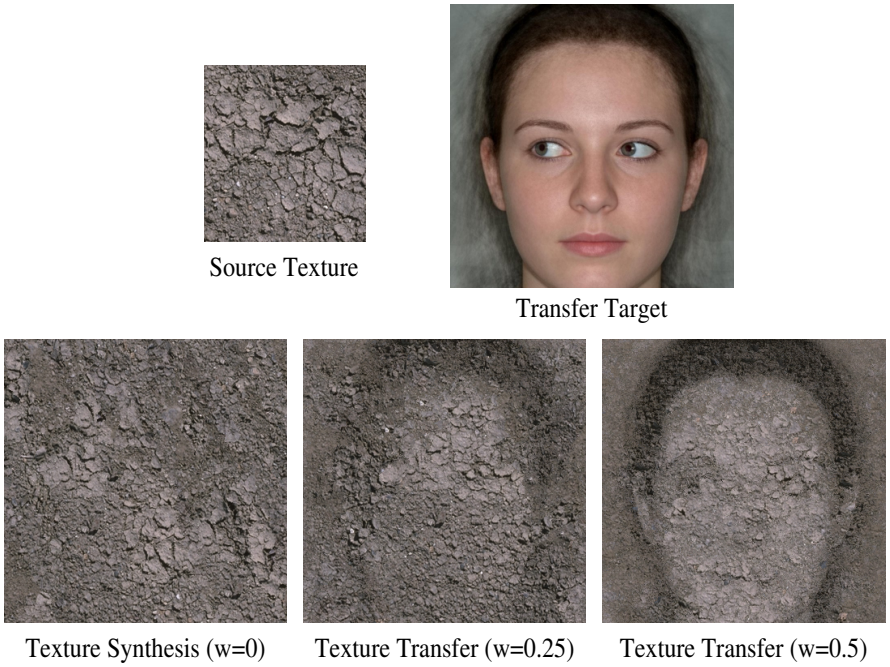


Figure 7.2: Texture transfer

lead to an impractical editing system. Furthermore, the approach is only suitable for textures which (at least roughly) satisfy the stationarity condition of Markov Random Fields.

A possible solution is provided by the Image Analogies framework of Hertzmann et al. [HJO+01] which aims at solving the following problem:

> Given a pair of images $\mathbf{A}$ and $\mathbf{A}'$ (the unfiltered and filtered source images, respectively), along with some additional unfiltered target image $\mathbf{B}$, synthesize a new filtered target image $\mathbf{B}'$ such that
>
> $$\mathbf{A} : \mathbf{A}' \quad :: \quad \mathbf{B} : \mathbf{B}'$$
>
> In other words, we want to find an "analogous" image $\mathbf{B}'$ that relates to $\mathbf{B}$ in "the same way" as $\mathbf{A}'$ relates to $\mathbf{A}$.

While this is a quite general problem which cannot easily be solved for many combinations of images, Hertzmann et al. proposed an elegant algorithm for solving the problem based on texture synthesis which works surprisingly well for many types of analogies. For our application of data-driven BTF editing the power of the formulation lies behind the introduction of the unfiltered source image $\mathbf{A}$

because now the constraint $\mathbf{B}$ does not need to be compatible with the source BTF ($\mathbf{A}'$ in this case) but only with the unfiltered source $\mathbf{A}$, which can be of significantly lower dimensionality than $\mathbf{A}'$. Hence, the image-analogy framework fits perfectly into our back-propagation based editing scheme (Figure 7.1) since it is able to propagate changes made in a simple version of a complex image back to the complex image.

A good example in this respect is the *painting-by-numbers* application also presented in the original paper. A simple "number-image" alone does not suffice to constrain the synthesis in a meaningful way (Figure 7.3, bottom), but if the original image is set in correspondence with an appropriately segmented number-image things work out quite well (Figure 7.3, middle). This example demonstrates that the apparent "intelligence" of the Image Analogies algorithm lies mainly in the correspondence between the complex filtered source texture $\mathbf{A}'$ and the less complex unfiltered source texture $\mathbf{A}$.

It is useful to imagine the Image Analogies framework as an extended constraint texture synthesis algorithm which extends the source texture by additional channels represented by the unfiltered source texture $\mathbf{A}$. During synthesis only these additional channels are fully constrained (by the unfiltered target texture $\mathbf{A}$) while the other channels are synthesized pixel-by-pixel using a causal neighborhood. Thereby, the algorithm balances between preserving the structures of the source textures and following the user-defined constraint.

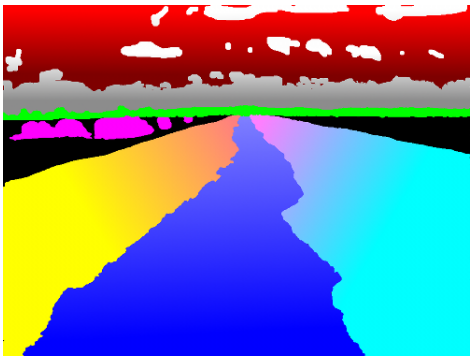
This scheme is reflected in Algorithm 2 which enlists the simple pixel-by-pixel version of Hertzmann's method (without multi-level synthesis).

Algorithm 2 Pixel-wise Image Analogy

input unfiltered source and target images $\mathbf{A}, \mathbf{B}$, filtered source image $\mathbf{A}'$

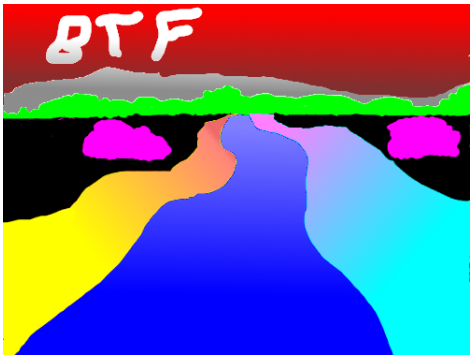
initialize search acceleration structures for $\mathbf{A}, \mathbf{A}'$
for each pixel $\mathbf{p}$ in $\mathbf{B}'$ **do**
  $N_{\mathbf{B}}$=neighborhood($\mathbf{p}$,$\mathbf{B}$)
  $N_{\mathbf{B}'}$=neighborhoodCausal($\mathbf{p}$,$\mathbf{B}'$)
  $\mathbf{p}_S$=findBestFittingNeighborhood(concat($N_{\mathbf{B}}$,$N_{\mathbf{B}'}$),$\mathbf{A}$, $\mathbf{A}'$)
  $\mathbf{S}(\mathbf{p}) = \mathbf{p}_S$
end for

return Source coordinate map $\mathbf{S}$



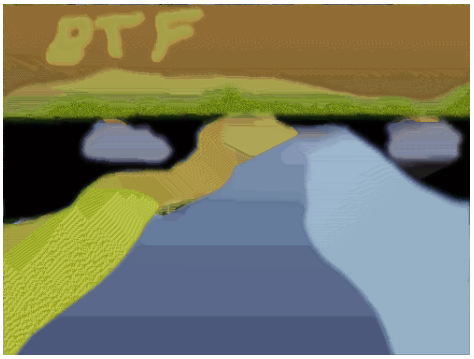
Unfiltered Source $\mathbf{A}$ (number image)

Filtered Source $\mathbf{A}'$(original image)



Unfiltered Target $\mathbf{B}$

Filtered Targeted $\mathbf{B}'$



Result using simple texture transfer

Figure 7.3: Painting-by-numbers is an illustrative example for the power of the image analogies framework. By creating a correspondence between the simplified source image $\mathbf{A}$ and the original image $\mathbf{A}'$, editing of the complex original can be performed via editing $\mathbf{B}$ and propagating the results. Simple texture transfer like in Figure 7.2 fails here because the algorithm can not do better than trying to fit the colors.

CHAPTER 8

BTF-ANALOGIES

The BTF-Analogies technique being presented in this chapter can be considered a prototypical instance of the back-propagation based editing scheme introduced in Chapter 7. The feature-of-interest model is represented by an extractor of spatially varying features. Since the features are extracted from the original data the feature map and the per-texel reflectance functions are in spatial correspondence automatically. This correspondence is exploited to propagate the editing operations applied to the feature map back to the BTF. We have termed this approach BTF-Analogies because it can interpreted as a generalization of Hertzman's Image Analogy framework to BTFs.

Figure 8.1 illustrates the components of our technique which will be detailed out in the following sections. According to this illustration the first version of our BTF-analogy system would be the following (compare with Figure 8.1):

Unfiltered source image $\mathbf{A}$	Feature map extracted from the BTF
Filtered source image $\mathbf{A}'$	Original BTF
Unfiltered target image $\mathbf{B}$	Edited feature map
Filtered target image $\mathbf{B}'$	Edited BTF which reflects the feature edit

While this sketch sounds reasonable there are several problems to solve until such a system would become practical. The first point is that the full complexity of the BTF is involved in the synthesis process which would slow down the algorithm prohibitively. The second problem is the feature extraction: what kind of features are suitable for editing and how can they be extracted automatically? The following subsections will be dedicated to these issues.

8.1 Multi-Channel Image Analogies

In principle the basic Image Analogies algorithm works for images with arbitrary numbers of channels. In practice the channel number affects mainly the dimen-

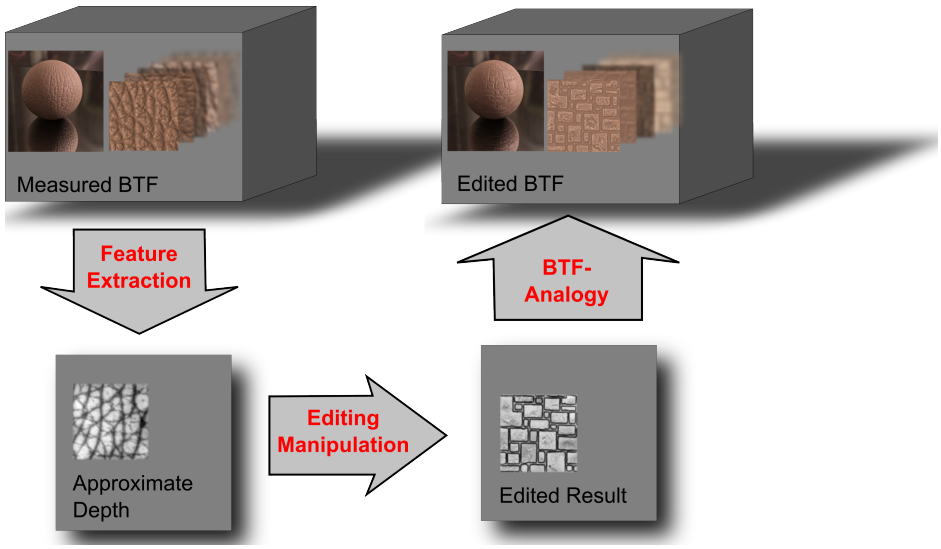


Figure 8.1: BTF-Analogies: propagation of editing operations applied to low-dimensional feature map to the full BTF.

sionality of the neighborhood vectors $N_{\mathbf{B}}$ and $N_{\mathbf{B}'}$, which grows linearly with the number of channels. For a spatial neighborhood vector $\mathbf{n_p}$ with a neighborhood radius of r texels and a number of texture channels c, we have for L-shaped (causal) neighborhoods

$$\mathbf{n_p} \in \mathbb{R}^d \text{ with } d = \lfloor (2 * r + 1)^2 / 2 \rfloor * c$$

In case of measured BTFs the number of channels corresponds to the number of images times the number of color channels (around 20.000 to 70.000 for the examples used in this thesis). This results for a typical radius of $r = 2$ in neighborhood vector sizes of 240.000 to 840.000 dimensions (the multi-level version of the algorithm would require even larger vectors) which is obviously infeasible. These vector spaces are far too large to perform an efficient search for the best-fitting neighborhood, which forms the basis of the Image Analogy algorithm. Hence the number of channels of the BTF has to be reduced *before* building up the neighborhood vector space. Otherwise, the neighborhood search would be heavily under-constrained and practically it would not even be possible to construct a search acceleration data structure for the neighborhood vectors because all neighborhood vectors would need a few hundred gigabytes of storage (e.g. 215 GB for a 256x256 texel BTF with 151x151 images and float precision).

An elegant solution to this problem is the so-called *Appearance Space* [LH06]. The idea is to perform a dimensionality reduction of the neighborhood vector

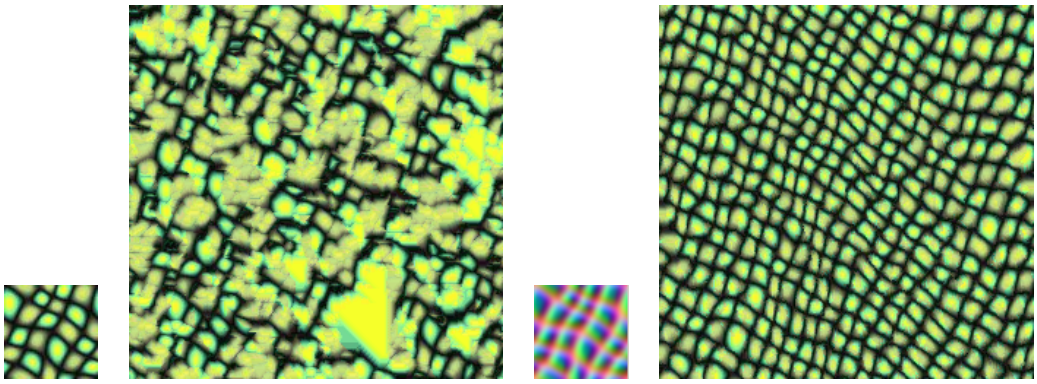


Figure 8.2: Appearance space texture synthesis - both enlarged result images were computed using k-coherance based neighborhood vector search [TZL+02] with neighborhood radius $r = 1$. On the right the appearance space texture (first 3 channels are shown) was used which drastically improved synthesis quality for almost equal runtime costs.

space and to perform the synthesis within this space. The technique works as follows:

Let $\mathbf{D_n} = (\mathbf{n_p})_{\mathbf{p}\in[1...w]\times[1...h]} \in \mathbb{R}^{d\times(w\cdot h)}$ be the neighborhood data matrix of a texture T_c with extend $w \times h$ and a number of c channels per texel $\mathbf{p}$ and d given as above. Standard pixel-wise texture synthesis can now be interpreted as searching for a column of $\mathbf{D_n}$ which has minimum L_2 distance to the neighborhood of the current texel in the target texture. As sketched above this search is quite expensive even for RGB-color textures and becomes impractical for many-channel textures like BTFs. One solution is to apply PCA to $\mathbf{D_n}$ and perform dimensionality reduction by keeping only $k \ll d$ non-zero eigenvalues: $\bar{\mathbf{D}}_\mathbf{n} \approx \mathbf{C}\mathbf{W}^T$ where $\bar{\mathbf{D}}_\mathbf{n}$ is the mean-centered neighborhood data matrix and $\mathbf{C} \in \mathbb{R}^{d\times k}$ and $\mathbf{W} \in \mathbb{R}^{(w\cdot h)\times k}$ are the PCA weight and component matrices. The trick is now to interpret the weight matrix $\mathbf{W}$ as a k-channel texture T_W and to perform textures synthesis for this texture instead of T. Since now one texel $\mathbf{p}$ of T_W essentially contains the information of a full neighborhood of T, very small neighborhoods (typically $r = 1$) suffice for a high-quality synthesis of T_W and therefore also of T.

As illustrated in Figure 8.2 the technique significantly increases the quality of standard texture synthesis for almost the same runtime cost. Another motivation for the technique is its ability to further improve synthesis quality by "enriching" the texture by additional "semantic" channels like edge feature maps which help to preserve the features with negligible additional run-time cost. In the same way it is applicable for synthesizing higher dimensional texture like BTFs.

The only additional problem coming up is the size of the neighborhood vector

data matrix which reaches hundreds of gigabytes as sketched above. In order to avoid building up the full matrix $\mathbf{D_n}$ we propose to use one of three different heuristics which – from our experience – yield comparable results:

1. Apply dimensionality reduction, e.g. PCA, to the many-channel texture before building up the neighborhood space.

2. Use only images from the frontal view (and apply dimensionality reduction to these images).

3. Greedily select a set of images from the data which are representative for the sample. Alternatively, the algorithm recently presented in [FCGH08] could be used.

Since the appearance space is such a useful extension to texture synthesis we will usually compute appearance space textures also for the unfiltered source and target feature textures. Therefore, the BTF analogies algorithm can be formulated as follows:

Algorithm 3 Pixel-wise BTF Analogy

input unfiltered source and target feature images $\mathbf{F}_a, \mathbf{F}_b$, BTF $\mathbf{G}$

$\mathbf{A}$=appSpaceTexture($\mathbf{F}_a$)
$\mathbf{B}$=appSpaceTextureProj($\mathbf{F}_b$)
$\mathbf{G}_{red}$=btfDimReductionHeuristic($\mathbf{G}$)
$\mathbf{A}'$=appSpaceTexture($\mathbf{G}_{red}$)
$\mathbf{S}$ = **Algorithm 2**($\mathbf{A}, \mathbf{B}, \mathbf{A}'$)

return Source coordinate map $\mathbf{S}$

Please note, that both $\mathbf{G}_{red}$ and $\mathbf{A}'$ need to be computed only once for each BTF. $\mathbf{A}$ needs to be computed only once per feature per BTF which means that it can be precomputed for each BTF and feature type combination. Therefore, during editing we only need to project $\mathbf{F}_b$ on the appearance space of $\mathbf{F}_a$, which is quite cheap, and then perform Algorithm 2, which is the most costly run-time part of the BTF analogy algorithm. If the multi-level version of the algorithm is used, the input pyramids $\mathbf{A}$ and $\mathbf{A}'$ and also the neighborhood search acceleration data-structure which consists of the concatenated neighborhoods from $\mathbf{A}$ and $\mathbf{A}'$ can also be pre-computed. In our implementation we use the k-coherance search [TZL+02] as acceleration strategy which leads in our un-optimized and unparallelized prototype implementation to synthesis times of about 2-4 seconds for a texture of 256x256 texels in size (Intel E6400 Core Duo with 3 GB RAM). This

is already not far from being interactive and an optimized version of the algorithm will be undoubtedly even more responsive.

8.2 Features for BTF analogies

In this section we will discuss two different types of features that can be used for BTF editing based on BTF-Analogies. The simplest feature will be a single color image that can be chosen from the original BTF data. As we will see in the following, only relatively simple editing operations can be successfully propagated back to the full BTF this way. Especially maintaining consistent shading effects for complex materials is difficult. Therefore, we also demonstrate editing based on reconstructed 3D information, i.e. a depth map. Using this more elaborate feature map, shading effects will be preserved more faithfully.

8.2.1 Single RGB/Luminance images

For most materials a single image captures already many important features, especially if the reflectance behavior is simple and the meso-structure is approximately flat. In this case it can be expected that simple editing operations like delete or copy can be propagated well already by editing only a single RGB or luminance image.

8.2.2 Reconstructed height-map

BTFs are especially interesting for representing materials with a significant meso-structure, because shadowing, masking and higher order light-transport effects are difficult and costly to simulate. At least the first order light transport effects like shading, shadowing and masking which usually make up the major component of appearance are already well described by a depth map. This fact was already recognized by Liu et al. [LYS01] who synthesized BTFs using reconstructed meso-structure. Their main goal was to synthesize a continuous BTF from the sparse BTF samples offered in the CUReT database [DvGNK97]. To synthesize a novel view/light configuration they rendered the synthesized meso-structure with the corresponding viewing and lighting parameters and used the resulting image as guidance for copying appropriate sample blocks from the captured images. Thereby, they were able to enforce consistency across different view/light configurations, although they synthesized each image of the BTF independently.

Fortunately, due to the recent progress in BTF measurement methodology much more densely sampled BTF data than the original CUReT data is now

available. These datasets can be used without the requirement of synthesizing novel view/light configurations. Instead, we propose to use the reconstructed meso-structure as low-dimensional representation within the BTF-image analogies framework.

The first requirement here is to reconstruct the height map of the material. Reconstructing the meso-geometry of textures is an active research topic in its own right. Since our samples are only moderately glossy and our method does not depend on a perfectly accurate reconstruction we found classical photometric stereo techniques [RTG97] and normal integration [FC89] to be sufficient for our experiments. To increase the reconstruction quality for more specular materials it is possible to use suitable reconstruction techniques like that of Chen et al. [TCS06] or even our own data-driven technique from Chapter 5.

Figure 8.3 shows results of the photometric meso-structure reconstruction for some of the materials used in this thesis. Please note that the constraint synthesis method is robust against low-frequency errors typically present in photometric reconstructions since it is based on local feature similarity.

Computing Derived Features

During our experiments we discovered that additional features derived from the height-map help to improve the search for best-matching neighborhoods in many cases (see Figure 8.4). This is a well known issue of the L^2-norm which is commonly used for neighborhood comparison in texture synthesis (cf. [ZZV$^+$03] and [LH06] for a more in-depth discussion of this topic). The additional features we derive from the original depth map are normal vectors and view-dependent occlusion.

To compute the normal vectors we simply convert the depth map into a triangular mesh and compute the per-texel normals by averaging the normals of adjacent triangles.

Computing occlusions is a little bit more difficult because doing it straightforward, e.g. using ray-tracing would be far too costly for interactive editing. Therefore, we employ graphics hardware to rapidly compute occlusions via shadow mapping. Our implementation is able to process about 500 view directions per second for a 60k vertices mesh on a NVidia 8800 GTX. This sampling resolution typically suffices to reliably compute a 4th- or 5th-order Spherical Harmonics expansion of the view-dependent occlusion which is then used as feature map.

By using the appearance space technique these additional features can be incorporated into Algorithm 3 with negligible additional run-time costs by creating a multi-channel texture from the depth-, the normal- and the occlusion-map and computing the appearance space texture from it.

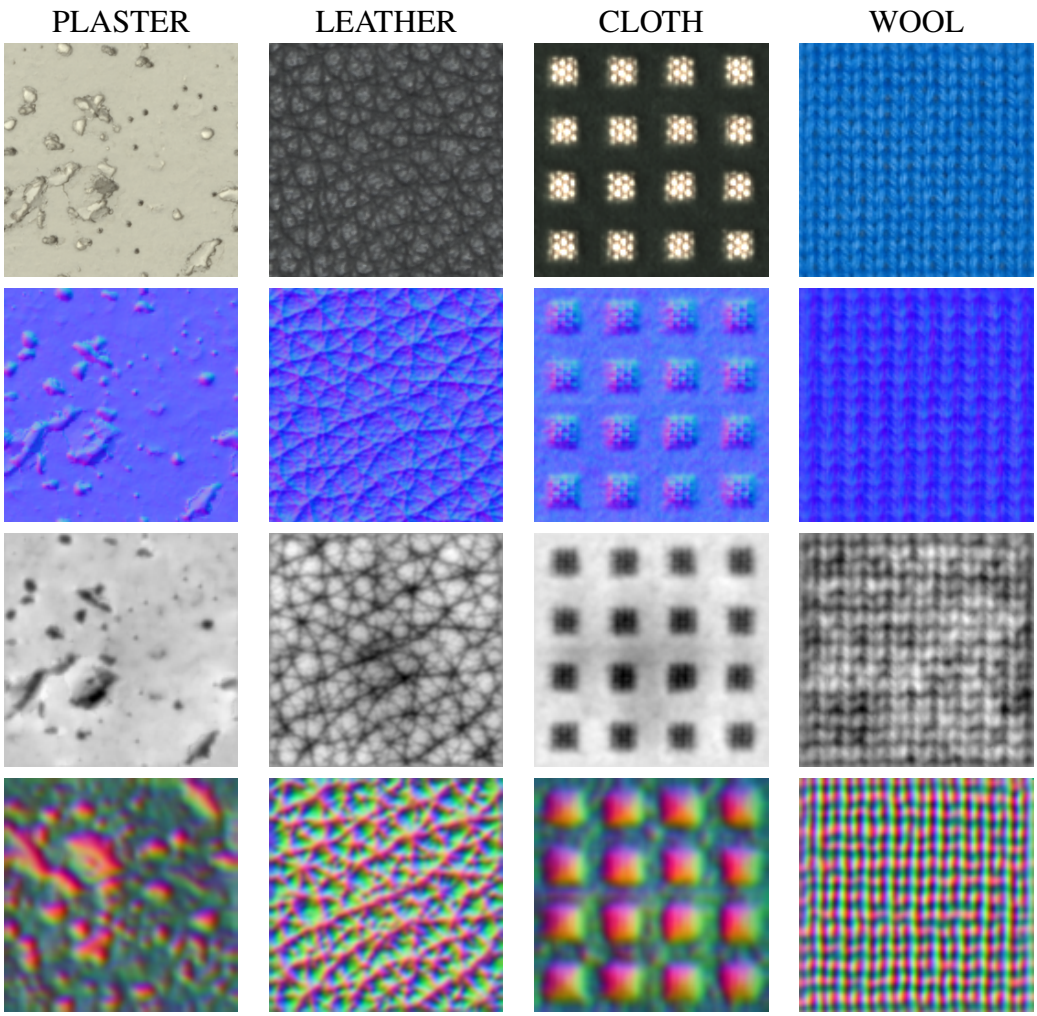


Figure 8.3: Meso-structure reconstructions of the materials used in this thesis. From to to bottom: *frontal view, normals, depth, appearance space*. All results where generated using photometric stereo and normal integration. The normal maps are shown using the standard RGB encoding. The appearance space image visualizes the first three PCA weights.

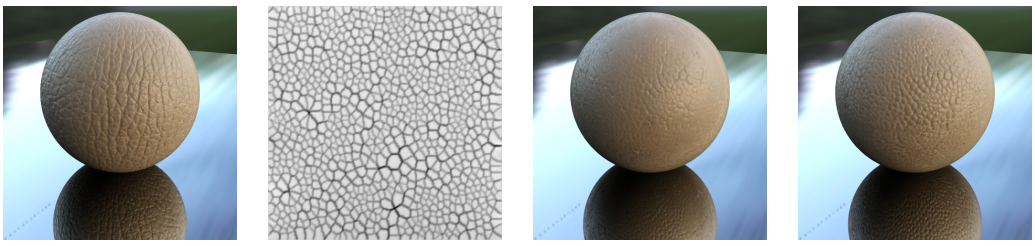


Figure 8.4: Using additional geometric features features improves BTF Analogies. From left to right: Original BTF, depth constraint, result using only depth feature, result using also normal vectors and view-dependent occlusion.

8.3 Experiments and Results

For testing the BTF-Analogy algorithm and the presented feature representations we created three different and practical editing scenarios and investigated, how well the editing operations are propagated to whole BTF depending on the material and feature types. We used the four materials depicted in Figure 8.3.

The results are shown for each material in Figures 8.5–8.8, and they are always presented as follows: For each material we show a rendering of the original material and the feature-of-interest maps which are either a frontal view image with averaged lighting or an approximate depth map. Furthermore, we show an image of the first three channels of the appearance space texture computed from the dimensionality reduced BTF (which should not be confused with the appearance space textures in Figure 8.3 which were computed from the approximate depth map and the corresponding features). We usually kept the first 64 PCA components and computed a 12D-appearance space texture using a neighborhood size of 7x7 texels. The results are then shown together with the edited feature maps.

8.3.1 Painting

This application is inspired by the painting-by-numbers example from the original Image Analogies paper. The idea is to use the well-known pipette and cloning tools to select values and regions from the feature-of-interest map and to paint with them a novel image. Due to the author's quite limited painting skills the task was chosen to be simple but expressive: a smiling stick-figure face.

As can be seen in the images both the rgb-image and the depth-feature control maps are suitable for transferring the editing operations to the BTF. The results achieved using depth maps are slightly superior in terms of 3D-structure preservation as proven by the results for the PLASTER and LEATHER materials. On the other hand the simple rgb-image performs better in preserving the weaving

pattern of the gold fabric part from the cloth. This is probably owed to the fact that the weaving pattern was not well captured in the depth map. The wool material shows another interesting effect here: For the rgb-image the high-frequency information is mainly lost at the location of the paint strokes but the geometry of the main threads is still preserved. For the depth-feature the algorithm tries to preserve also the high-frequency information of the sub-threads which renders the editing appear less prominent.

8.3.2 Warping

An editing operation which should be a slightly easier task for the algorithm is warping the feature-of-interest map. The idea is to take the FOI map and apply a warping operation which basically corresponds to a spatially varying translation of pixels and a resampling of the original data. These operations should easily be reproduced by the method because the neighborhood search should be able to track the warp translations.

For the creation our examples, which consist of a few swirls in each image, we used the IWarp-Filter from the open source image processing software GIMP[1]. As expected, for these operations the method produced more or less meaningful results even for the wool material. Again, the depth feature (especially when extended by the derived, additional normal vector and occlusion features) achieves results which are more consistent with the original BTF appearance.

8.3.3 Transfer

While the first two applications mainly consist of editing the original FOI map some experiments with completely different feature maps were made, too. This operation could be interpreted as some sort of texture-to-BTF transfer. For the results depicted here we used an image of cobblestones as transfer target texture. Of course, the features and statistics present in the target texture should be roughly related to the BTFs FOI map because otherwise the algorithm simply would not find enough meaningful matches. Such a failure case is the wool material which has a meso-geometry quite different from the cobblestones.

This kind of editing operation is of particular interest when it comes to material interpolation. This way features of, e.g. different leathers or textiles could be transferred between each other in order to allow a more meaningful interpolation as will be detailed out in the following Chapter 9.

[1] `http://www.gimp.org`

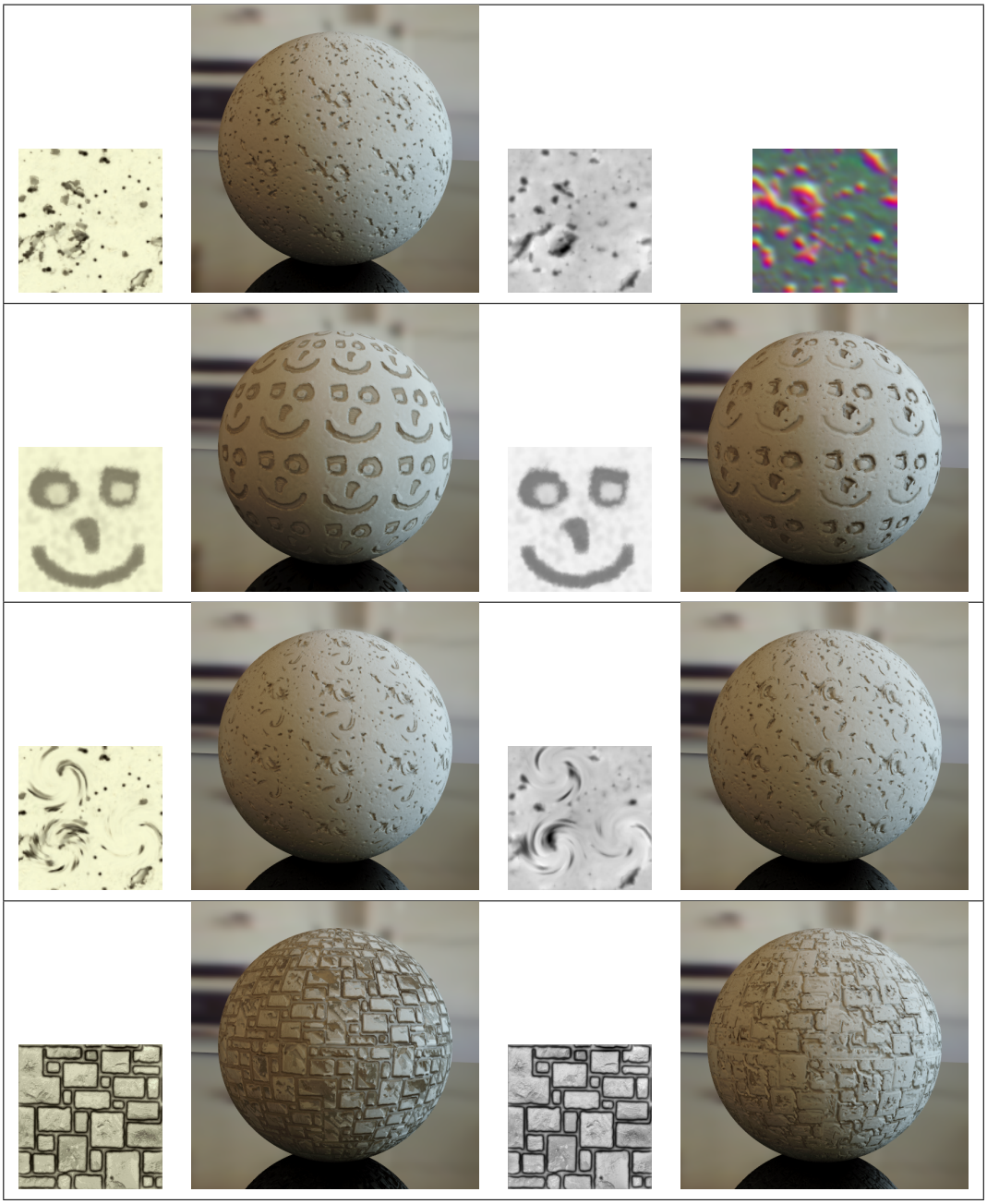


Figure 8.5: Editing results for the PLASTER material. Using depth as feature-of-interest generally improves the preservation of depth-related features. This effect is most notable for the transfer operation where the appearance of the scratches and valleys is preserved.

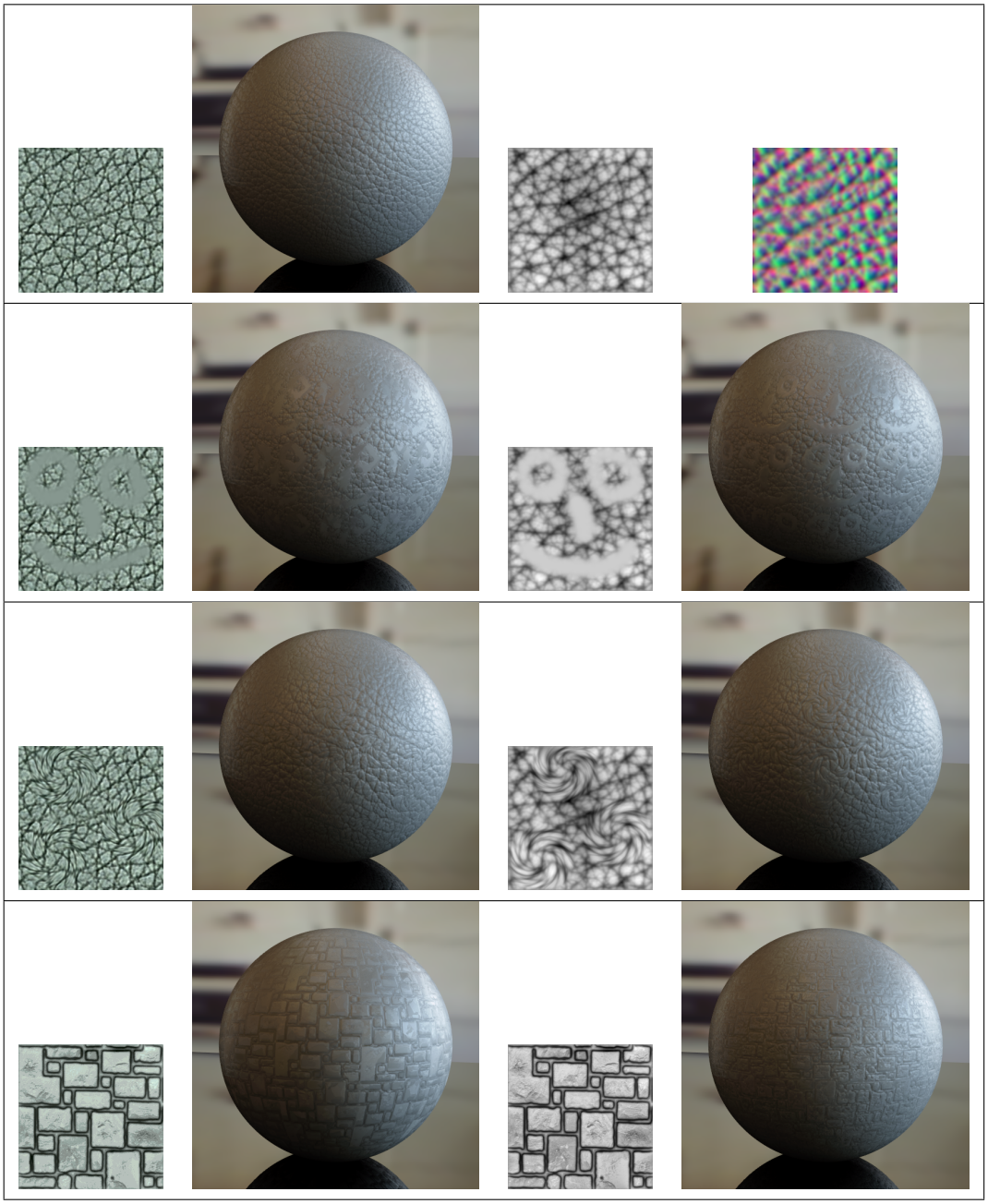


Figure 8.6: Editing results for the LEATHER material.

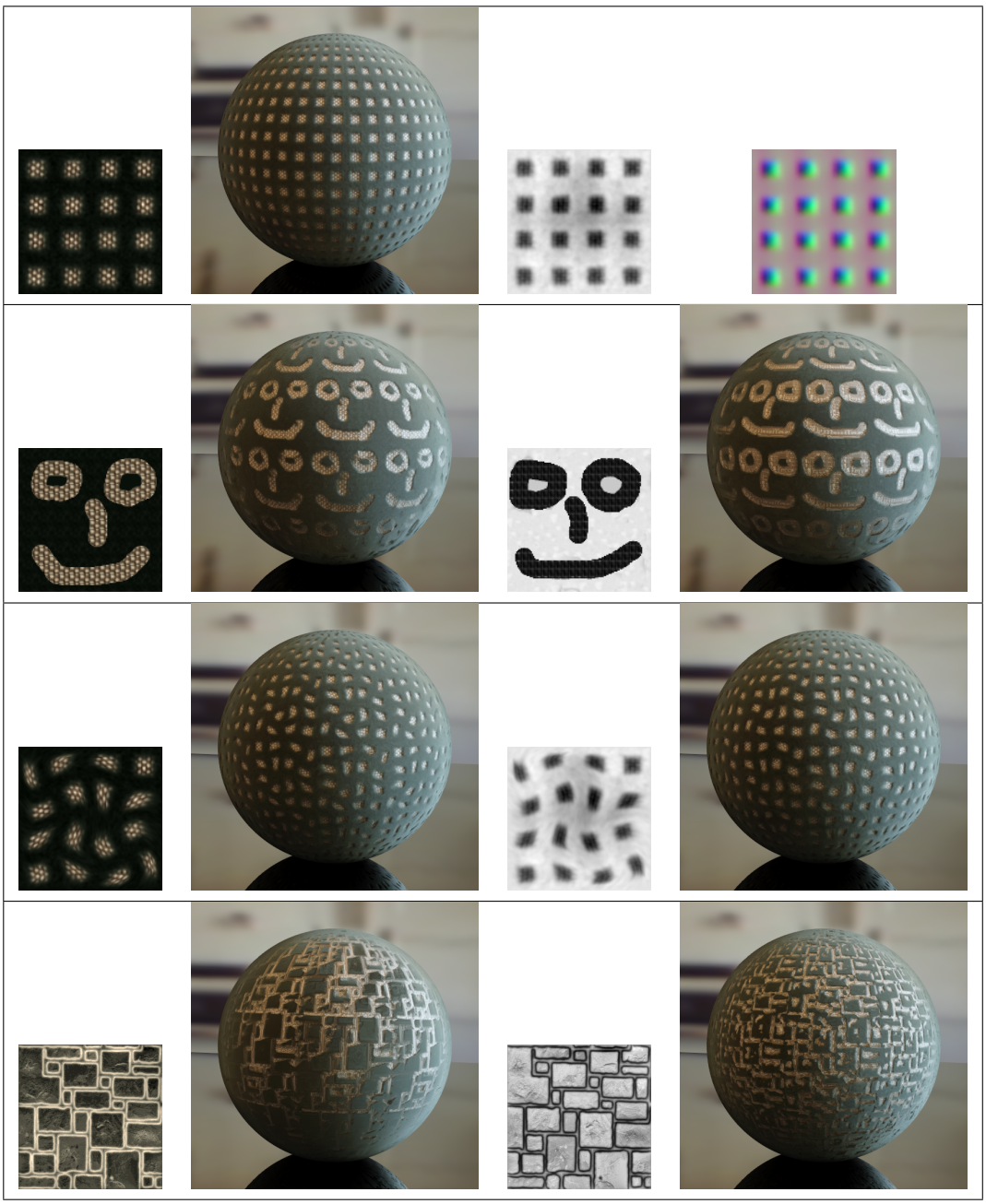


Figure 8.7: Editing results for the CLOTH material.

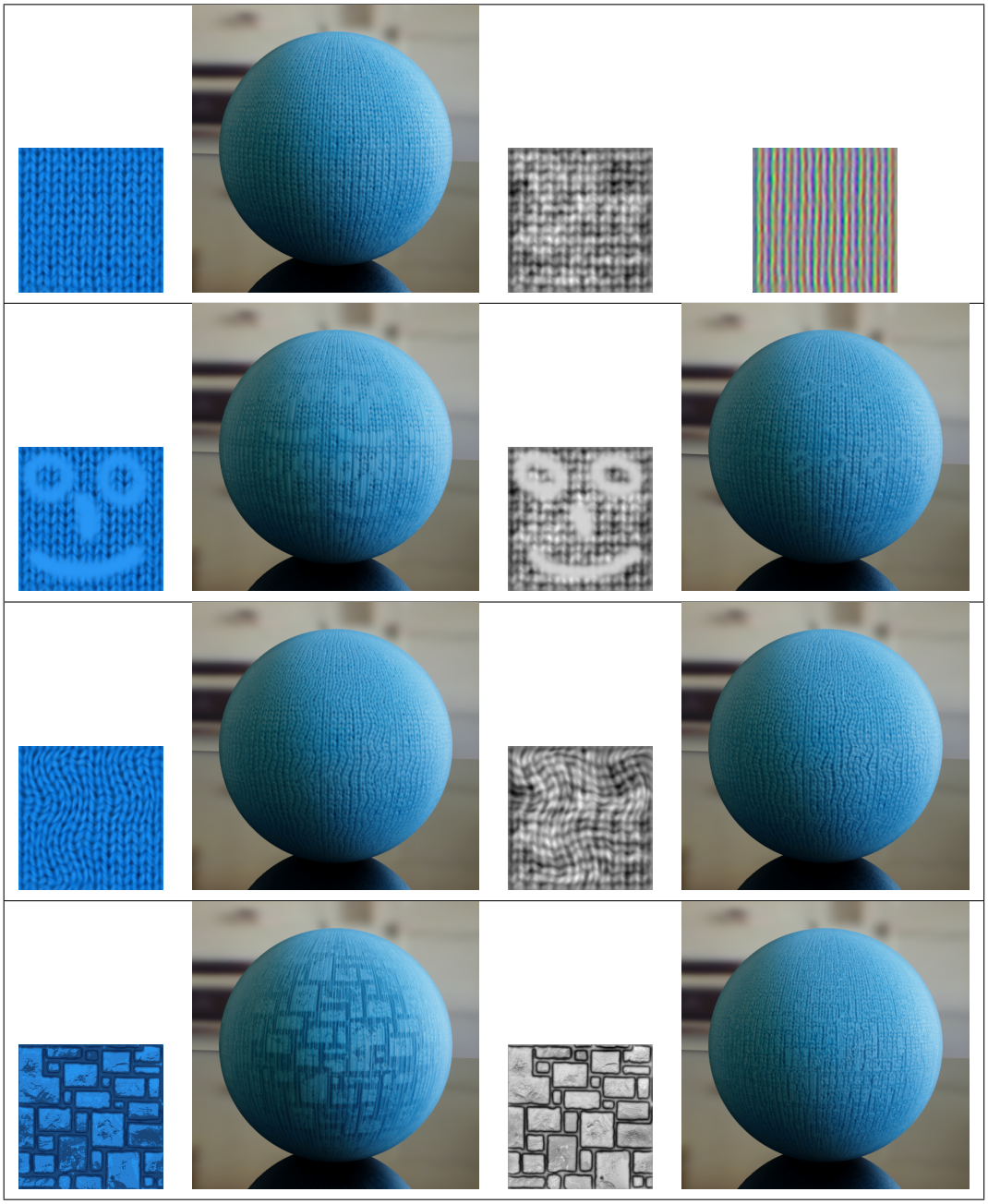


Figure 8.8: Editing results for the WOOL material.

CHAPTER 9

ANALOGY-DRIVEN BTF INTERPOLATION

The BTF Analogies technique introduced in the previous chapter is already a powerful tool to edit and manipulate spatially varying features of measured BTFs. However, applying the editing operations to achieve a desired goal can still be a difficult task and might require artistic skills. Furthermore, the technique does not allow to edit the micro-scale reflectance behavior of the BTF.

In this chapter we present *analogy-driven BTF interpolation* which provides solutions to these two problems. Based on the BTF Analogy algorithm this interpolation allows to change the micro-scale reflectance behavior of BTFs in a fully data-driven way. Furthermore, as illustrated in Figure 9.1, by generating a common feature-of-interest map using, e.g. a procedural texture model, new BTFs can be created by the manipulation of a few sliders only. The idea is to utilize the procedural model for the generation of the FOI map which then is used as input for the BTF Analogy. By employing the same FOI map for multiple BTFs the interpolation between these BTFs becomes meaningful since the important spatially varying features are all aligned to this single FOI map. Using this approach a hybrid procedural and data-driven model for a specific material class can be generated.

In the following we will detail out the efficient interpolation of multiple BTFs in Section 9.1. Then a procedural model for leather meso-geometry textures which exemplifies our idea of a hybrid procedural and data-driven model will be introduced in Section 9.2. The closure of this chapter consists of presenting experiments and results in Section 9.3.

9.1 Interpolating Multiple BTFs

A straight-forward implementation of Figure 9.1 is possible but resource intensive since the BTF Analogy algorithm is performed independently for each of the input BTFs which have to be kept in main memory. Since it is likely that the input BTFs are from the same material class it can be worthwhile to exploit inter-material cor-

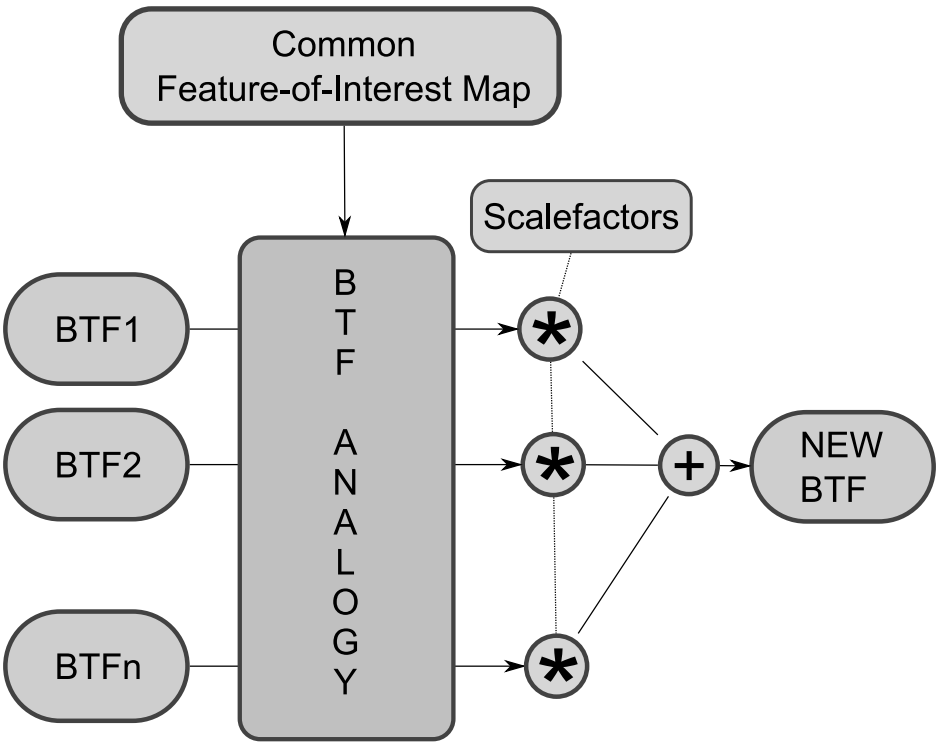


Figure 9.1: Meaningful BTF interpolation driven by BTF Analogies. The BTF Analogies are used to align the spatially varying features to a common feature-of-interest map, e.g. generated by a procedural model. Then these spatially aligned BTFs can be linearly combined with user-defined scale-factors.

relations. In fact, interpolating BTFs from completely different material classes like shiny metal and wool cloth might be funny but it does not make much sense for practical applications. It is more reasonable to combine materials from related material classes. In this case the spatially varying features are more likely to have similar statistics which improves the results of the BTF Analogy.

One possibility to exploit the inter-material correlations would be arranging all materials in a tensor with an additional *material* mode and to factorize this extended tensor. Considering the results from Chapter 4 we propose a different strategy based on block-wise factorization:

First, we factorize the data matrix of each material using standard SVD and keep the first 150 Eigenimages. This usually suffice to reduces the whole data to a size that fits into main memory. Then we perform an inter-material spatial clustering based on these Eigenimages. We typically used about 250 clusters. Then we factorize the original data in each cluster and keep about 8-10 eigenvectors per cluster to allow for fast evaluation during run-time.

Now, meaningful interpolation during editing can be afforded. For example, a set of perceptually meaningful directions in the spirit of the data-driven reflectance model from Matusik et al. [MPBM03] can be defined for the material class. Each material will be assigned a score along the particular trait vector (e.g. by performing a simple user study). These scores then determine the interpolation weights for the respective material when the user chooses particular values for the different traits.

During editing for each material in the database, which has been assigned a weight larger than a small threshold, we perform the meso-structure constraint synthesis. This returns a set of references into the BTF data which are used to lookup and reconstruct the respective reflectance values from the compressed database. Then these values are interpolated using the given weights. Results of our method are given in Section 9.3.

9.2 Procedural Leather Meso-Geometry

We will now exemplify our idea of a hybrid procedural and data-driven material model. Since the data-driven part of the model is represented by a set of BTFs and the BTF-Analogy technique, the only thing left is the procedural model which is intended to model the feature-of-interest map of a specific material class. In order to give interactive feedback during editing such a model should be efficiently computable. Furthermore, the possibility of fitting the model to the measurements is desirable in order to start the editing process from a measured sample. Last, but not least, it would be advantageous that the model supports smooth interpolation between different instances of the model in order to support smooth interpolation

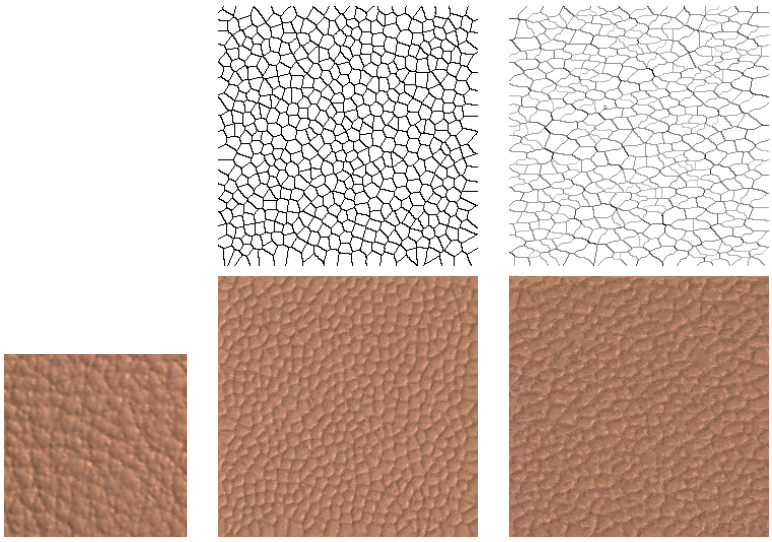


Figure 9.2: Left: Real leather texture. The analogy to a typical Voronoi crack pattern gives the typical "Voronoi-look". Our modified crack pattern with polyline based sites and depth variation offers a more natural variation of site shapes.

of different materials.

A model satisfying these requirements for leather-like materials will be presented in the following subsections. However, by defining appropriate features and corresponding procedural models we believe that our technique can be applied and generalized to many other interesting classes of materials like cloth or wood.

9.2.1 Procedural Voronoi Crack Patterns

There exist many algorithms which try to simulate biological or physical crack pattern generation processes resembling the appearance of typical leather crack patterns (the recent paper of Iben et al. [IO06] gives a good overview). Unfortunately, most of these methods are typically not interactive. Therefore, commercial leather shaders are often based on Worley's cellular texture basis function [Wor96] which generates cracks by computing the Voronoi diagram of randomly distributed sites and can be evaluated very efficiently. The drawback of this method is that the generated Voronoi sites have a not very realistic shape. Furthermore, we need a method to control the generation of cracks of different depths which is usually not support by these Worley shaders.

We propose two simple but effective solutions for these shortcomings. The first idea is to compute the generalized Voronoi diagram of polylines instead of

points. This allows to generate more realistically shaped sites. The second trick is to base the crack depth on the distance to the center of the neighboring sites.

The distribution of the sites can be controlled with arbitrary 2D-density functions. By default we use a randomly generated Perlin turbulence texture as density. The degree of randomness can be set by the user and controls the "spotiness" of the crack texture. The shape of the sites, i.e. their "roundness" and "anisotropy" can be controlled by setting average length, preferred orientation and regularity of the polylines. As illustrated in Figure 9.2 these modifications offer more variety in the shape of the sites and thus represent a more general class of leather-like materials.

9.2.2 Efficient Crack Pattern Computation

The computation of a crack pattern from the aforementioned parameters can be done very efficiently. We use the algorithm of Kopf et al. [KCODL06] to distribute the sites given a 2D-density. To compute the crack pattern from the sites we employ the GPU-based drawing algorithm for generalized Voronoi diagrams by Hoff et al. [HKL+99]. After assigning a depth to each crack we use a depth-dependent Gaussian crack profile to draw the crack heightfield into the z-Buffer. This process can be repeated for different scales and we allow to combine these scales with user-defined factors and to apply Gaussian filtering to each of the scales independently.

9.2.3 Morphing Procedural Crack Patterns

A great advantage of our procedural meso-geometry model is that natural morphs between different instances of the model can be computed easily. In our case this is not done by interpolating the model parameters but by finding correspondences between the sites. In particular we compute correspondences that minimize a morphing distance. Then we morph the structure by interpolating position and structure of the corresponding sites.

Our morphing distance is defined as follows: $A = \{a_i\}_{0<i\leq M}$ and $B = \{b_i\}_{0<i\leq N}$ shall be the polyline based Voronoi sites of two crack patterns with $M \leq N$. Then the morphing distance we try to minimize is simply

$$D(A,B) = min_f \sum_{i=0}^{M} d_\lambda(a_i, b_{f(i)})$$

where f is an injective mapping between $\{1 \dots M\}$ and $\{1 \dots N\}$. Finding the assignment f which minimizes D is a classical combinatorial optimization problem which can be solved efficiently using the Hungarian algorithm [Kuh55]. The

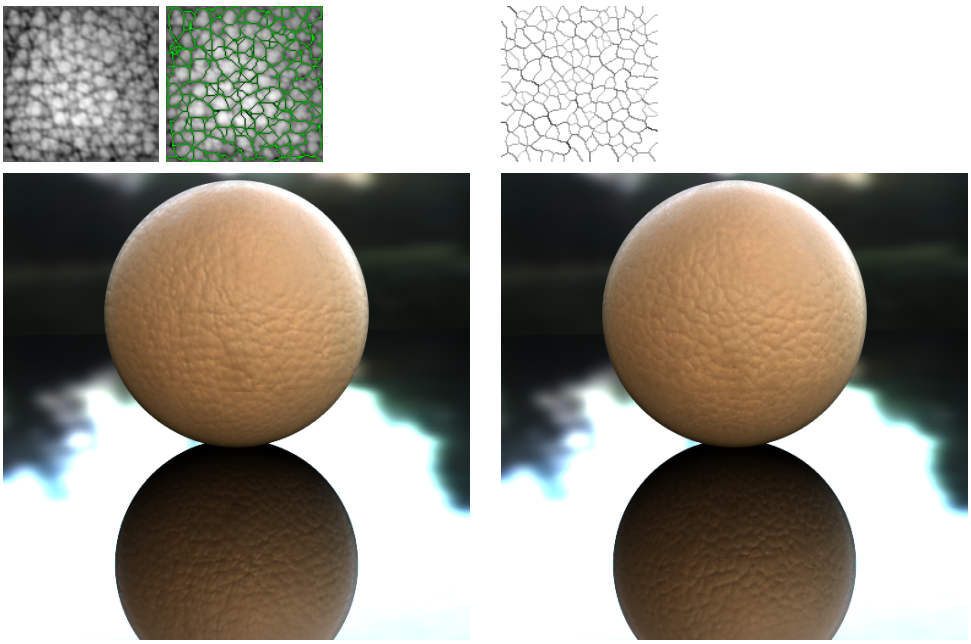


Figure 9.3: Fitting the procedural crack pattern. Left: Original BTF with depth map and cracks (in green) found by watershed algorithm. Right: Resulting Voronoi cracks and rendered result computed by the BTF-analogy algorithm.

distance d_λ between two sites a_i, b_j is defined as a weighted sum of the Euclidean distance between the centers of gravity $cog(\cdot)$ of the Voronoi sites and a shape distance $d_s(\cdot,\cdot)$ between the site borders of the Voronoi sites:

$$d_\lambda(a_i, b_j) = \lambda|cog(a_i) - cog(b_j)| + (1-\lambda)d_s(a_i, b_j)$$

There are numerous ways to define the 2D shape distance d_s.We used the approach of Sederberg and Greenwood [SG92] which is based on dynamic programming. This term can be used to enforce that the shapes of two sites that are blended together are roughly similar. Typically we choose λ close to 1 since a morph with a minimal movement of sites uses to be the visually the most pleasant.

The morphing itself is based on interpolating the centers of gravity of the sites and morphing the polylines themselves is also based on Sederberg and Greenwood. Since the number of sites typically does not match, the number of sites is equalized by generating new sites which are smoothly blended in during the morphing.

9.2.4 Fitting Crack Patterns to Meso-Geometry

Fitting the crack pattern texture to real meso-geometry is desirable because it allows representing real measured materials completely within our model and thus to start editing this representation from a real material.

In the case of crack patterns the fitting can be reduced to a computer vision task. First we detect the cracks in the meso-structure heightfield using a variant of a watershed image segmentation algorithm [RM00]. Then we compute a medial axis transform of the found crack pattern to determine the corresponding Voronoi sites. Since we currently only allow simple polylines as Voronoi sites the process tends to loose a bit of information during this step. As illustrated in Figure 9.3 this procedure reconstructs the overall shape and distribution of cracks and sites well. Comparing the rendered images closely a slight loss of high-frequency information can be spotted in the reconstruction. Apparently, this kind of information cannot be captured by a simple crack pattern.

9.3 Experiments and Results

In order to test our analogy-driven BTF interpolation we set up a database fifteen different real and imitated leather materials shown in Figure 9.4. Again, all these materials have been measured at the University of Bonn. The measurements have been cut to a spatial resolution of 128x128 texels. Since about 20000 HDR images (RGB) are captured per material the uncompressed storage size of this database is about 30 Gigabytes. Using BTF compression (adaptive block-wise factorization) the storage requirements for each material can be reduced to about 5-10 megabytes leaving us with 100 Megabytes for the whole database which is already manageable. We can reduce these requirements by an additional factor of more than 50% without reducing visual quality by exploiting inter-material correlations as proposed in Section 9.1.

A comparison between our analogy-driven interpolation and naive interpolation is shown in Figure 9.5. The bottom-left image shows the result of naive interpolation between three leather BTFs and a plaster BTF by interpolating each per-texel reflectance function. The naive interpolation results in a blurred meso-structure which does not appear very realistic. The bottom-right shows the result of interpolating the same BTFs but only *after* the spatial alignment to a common meso-structure using the BTF Analogy technique. The common meso-structure was generated using our procedural model for crack patterns introduced in Section 9.2. The interpolated result shows a much sharper and more realistic meso-structure while the micro-scale reflectance appears as a quite interesting and meaningful mix of the different materials.

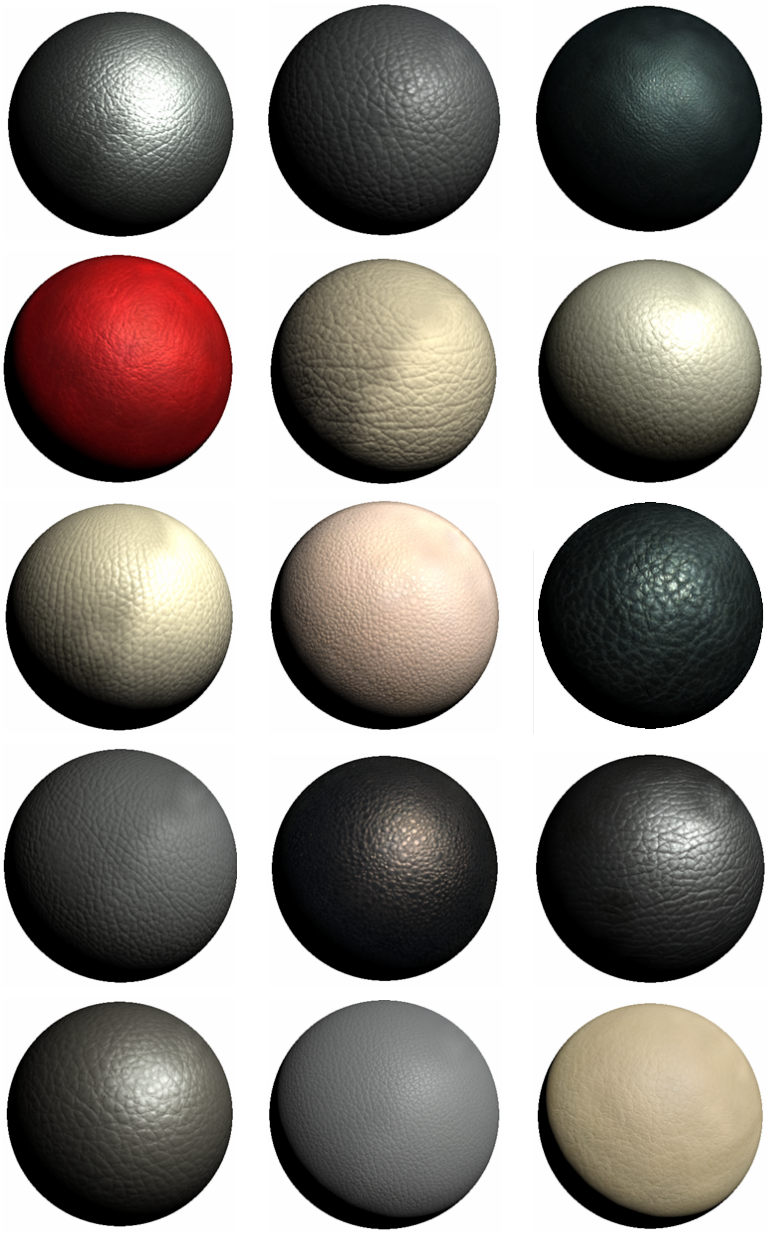


Figure 9.4: Fifteen different leather-like materials.

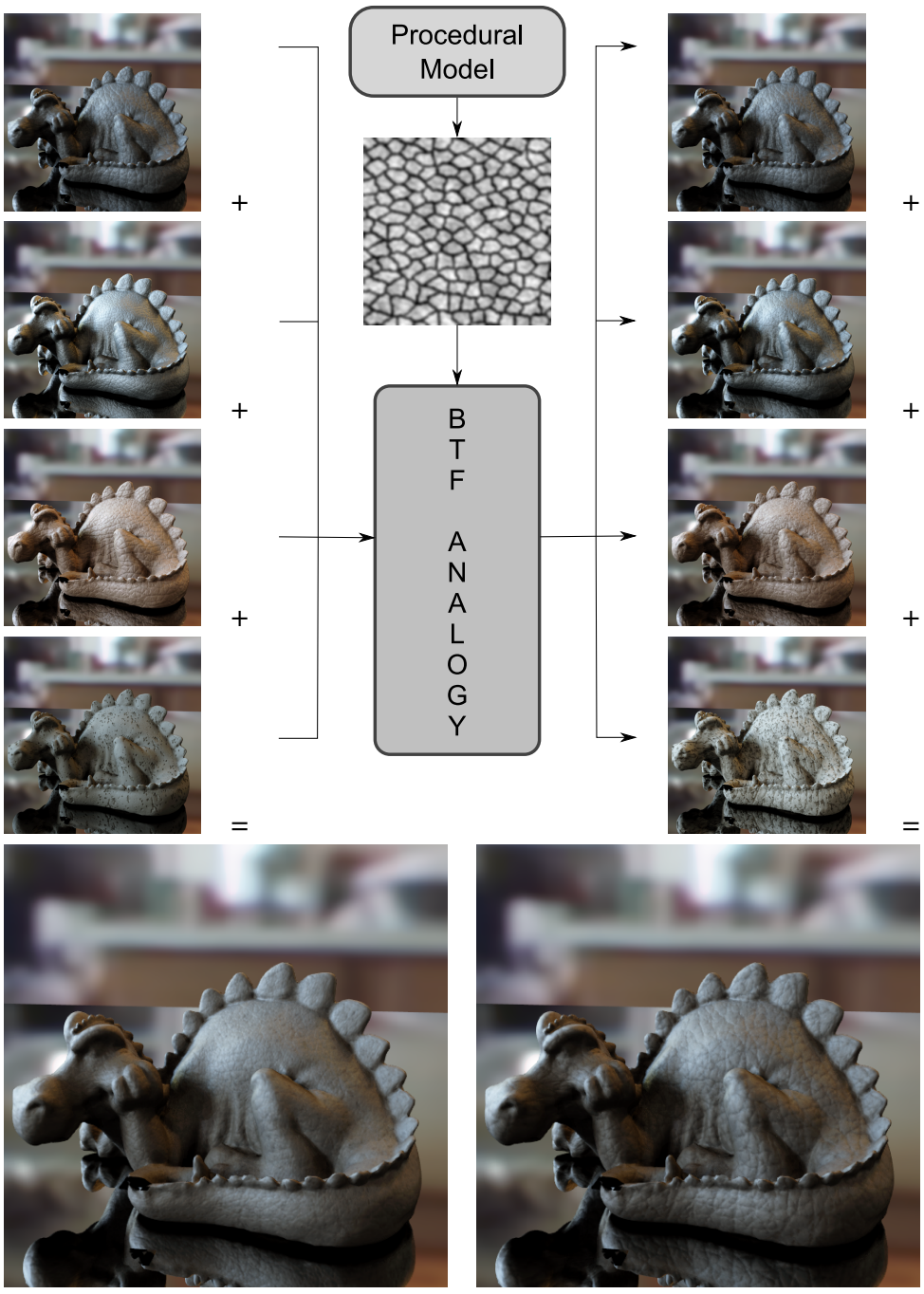


Figure 9.5: Naive interpolation of the different leather BTFs results in a blurred meso-structure (left). Using BTF-Analogies to create BTFs with similar meso-structure enables a more meaningful BTF interpolation.

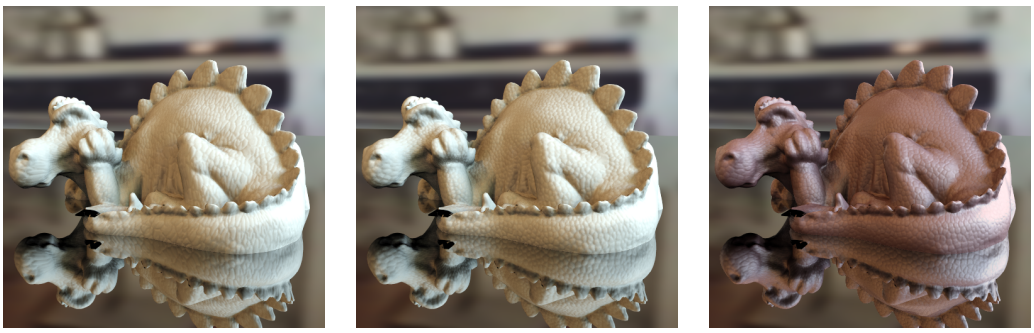


Figure 9.6: Editing of the "dragons skin". Left: Measured BTF, middle: making the meso-structure more regular, giving it a more spotty, lizard-like look. Right: increasing glossiness and shifting color makes the dragon look a little bit more evil.

An example for editing a leather BTF using characteristic material traits is shown in Figure 9.6. The original material is made more "specular" and "reddish" by mixing in materials which have been assigned high scores along these traits. Furthermore, the leather is made more regular by changing the parameters of the procedural model.

Finally, Figure 9.7 shows an example of a smooth morph between two materials from the database. It smoothly interpolates the structure and reflectance of the two leathers. Please take also a look at the accompanying video which gives an even better impression of the naturalness of the computed warp.

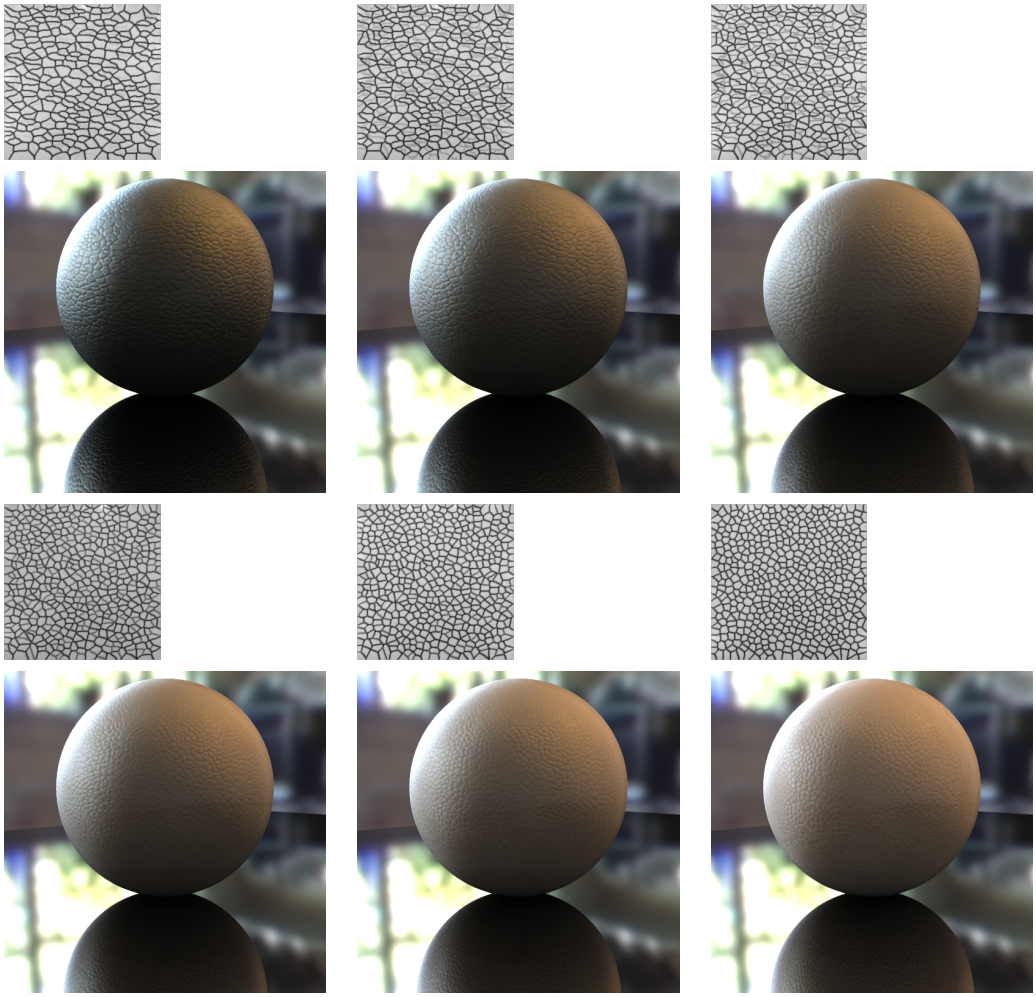


Figure 9.7: Morphing between gray leather and a beige imitated leather with regular structure. The morphed constraint texture is shown above each rendered image. The reflectance is interpolated and rendered from the measurements stored in the compressed BTF database.

CHAPTER 10

RELATED AND CONCURRENT WORK

The editing of high-dimensional measured appearance is a relatively new research area but it is based on a great deal of work from the fields of appearance modeling as well as texture synthesis and transfer from which we will only mention the most related work.

10.1 Appearance Modeling and Editing

Measuring reflectance and fitting analytical models to the data as introduced by Ward [War92] has a long time tradition in computer graphics. Such a representation can easily be edited by changing the parameters of the analytical model. Matusik et al. [MPBM03] questioned this measure-fit paradigm by claiming that analytical reflectance models only describe a restricted class of materials and that their parameters often have no intuitive physical meaning. Consequently, they proposed a data-driven reflectance model based on a large collection of measured BRDFs and applied data (manifold) analysis and perceptual studies to find perceptually meaningful directions in the space of these measured BRDFs. While such an approach seems to be feasible in the case of BRDFs no analogue model has been proposed for spatially varying reflectance yet.

Instead, different approaches suitable for a single material have been taken. Weyrich et al. [WMP+06] for instance proposed a measurement-based reflectance model for human skin based on the Torrance-Sparrow BRDF model. The spatial variation of the skin is captured in simple albedo maps which can be transferred between faces using the histogram equalization technique of Heeger and Bergen [HB95]. Unfortunately, such an approach is infeasible for materials with depth variation because in this case analytical reflectance models perform poorly [MMK04a]. The non-parametric Inverse Shade Tree approach by Lawrence et al. [LBAD+06] allows editing of measured spatially varying materials but the used low-term factorizations again restrict the method to flat materials which can be described by a few basis BRDFs only.

The first method dealing with non-flat materials is by Kautz et al. [KBD07] who introduced *BTFShop*: photo-editing-like manipulation of BTFs. Among the

proposed editing operators are tone and color editing using differential painting and change of color distribution, angular blur and sharpening for editing specularity and also quite general tools like *Copy & Paste* respecting parallax and shadowing. Some of these operators like the angular blur or the tonal operators are heuristics that can be used to mimic modifications of the micro-geometry and are applied to the whole data after the user has set their parameters. Other operators like color painting generalize basic tools known from image editing by propagating them from the top view to all other images using, e.g. differential edits. In order to account for parallax between different views in this case the authors define warp/unwarp operators based on an approximate geometric reconstruction. The system operates always on the full BTF data, i.e. edits are applied by modifying the raw per texel data, and it achieves interactive performance by employing a sophisticated out-of-core data management.

When it comes to the editing of the spatial distribution of textural features BTFShop has the same difficulties as traditional image editing software. For example changing the pattern of a stone wall using paint tools and cut & paste operations is rather cumbersome. Also editing the reflectance towards a specific existing material is rather difficult. Since BTF analogies and data-driven BTF interpolation are specifically designed for these tasks it is easy to imagine that our technique can be nicely integrated within BTFShop as an additional editing operator orthogonal to the ones presented in the paper.

Another technique orthogonal to ours is the AppWand algorithm by Pellacini and Lawrence [PL07]. It is designed to spatially propagate edits of reflectance properties by offering a generalized wand tool which selects regions of similar appearance. The tool behaves analogously to an image wand tool which selects regions of similar colors. The edit is actually performed by offsetting and scaling a desired parameter value for the samples selected by the wand tool and then the method propagates the edits consistently to regions of similar appearance. Thereby, it is mainly suitable for flat materials which can be represented by spatially varying parametric BRDF models. A more general formulation of the editing propagation objective function which does not a rely on a sparse neighborhood graph has been recently published by An et al. [AP08].

These methods are designed to consistently propagate parametric edits across complex spatial patterns. They are not suitable for editing spatial distributions of textural features as it is possible with our method.

10.2 Texture synthesis and transfer

The inherent complexity of texture has prevented the development of a general parametric texture model. Instead models for special types of textures have been

invented. Procedural models [EMP+94] are suited for natural textures like wood or stone. Parametric models based on the Markov Random Field model (e.g. [HF07]) are only capable of reproducing mainly stochastic textures and are expensive to evaluate. In either case the selection of appropriate parameters might be non-trivial and many textures can not be reproduced using procedural models. Therefore, researchers started to focus on non-parametric models based on example textures [EL99, WL00]. The idea is to generate a larger but perceptually similar version of a small texture sample by iteratively choosing pixels or patches from the sample which "fit well" (in terms of a specific metric) into the already synthesized part of the output texture.

While apart from avoiding repetition artifacts the generation of larger versions of input textures might not sound that useful in itself the significance of texture synthesis for material editing became apparent with the idea of constrained texture synthesis [Ash01, HJO+01]. In this case constraints, typically given as images or flow fields (e.g. [KEBK05]), are used to guide the synthesis process and thus enable some kind of user control for texture synthesis. The question remains how to generate these constraints. Zhou et al. [ZDW+05] presented a method which uses constraint graph-cut synthesis to allow the interactive placement of BTFs on surfaces. The synthesis is used for seamless blending and not for editing the BTF itself. In the work of Mertens et al. [MKC+06] a method is presented that transfers the texture of a scanned object to arbitrary geometry which then acts as the constraint.

Part III

Closure

CHAPTER 11

CONCLUSIONS

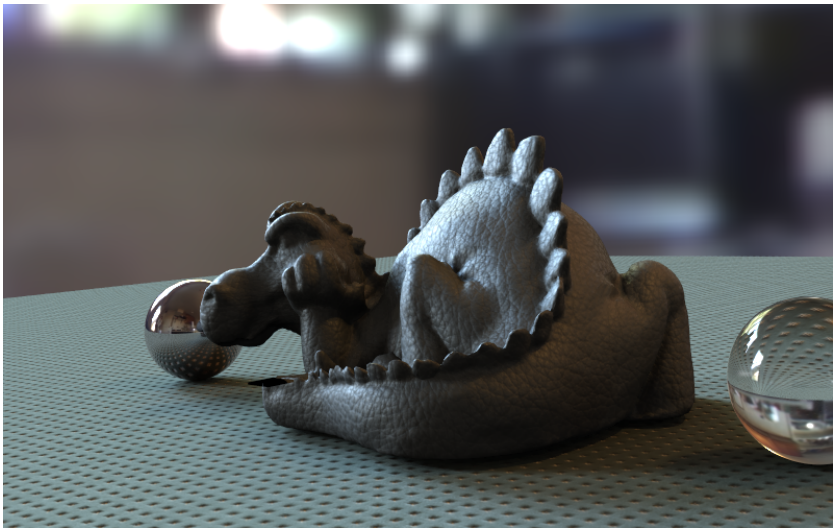


Figure 11.1: Rendering a leather dragon based almost completely on data-driven techniques: The lighting has been captured using a light probe and the geometry has been scanned. The materials on the dragon and the plane have been measured, compressed and edited by the techniques presented in this thesis.

11.1 Summary

The work presented in this thesis was driven by the dramatic improvements in digital sensor technology which now allow to capture gigabytes of visual data in an ever shorter time. Image-based data is able to represent the appearance of objects and materials with a fidelity and effectiveness almost impossible to achieve

with previous methods but it also demands for new techniques to handle the large amount of data.

Although the two different parts of this thesis can be comprehended more or less independent from each other they both serve a single purpose: making the application of complex measured appearance representations like the BTF more practical. From a personal point-of-view the advent of these data-driven techniques in scene modeling allows artistic dilettantes like the author of this thesis to create almost photo-realistic images like Figure 11.1 in only a few seconds. It follows the individual summary of each of the two parts:

Part I: Compression

The first part covered the most important component of any image-based rendering system: compression. Since the raw measurements from a single material alone can require the whole main memory of a today's personal computer without effective data compression the whole approach would be in vain. Such compression techniques should not only reduce the required memory to manageable dimensions while preserving the visual quality but also allow to reconstruct the original data fast enough as required for interactive and high-quality rendering applications.

In Chapter 4 we compared in this respect several compression techniques based on matrix and tensor factorization. We discovered that the factorization of adaptively computed matrix blocks offers the best compromise between compression ratio, reconstruction error and run-time costs. In our experiments we reduced the storage requirements of our examples from around 2-3 gigabytes down to 10-20 megabytes while preserving a high visual quality and reconstruction performance. We also showed that recently introduced compression techniques based on tensor factorization are currently not an option for practical applications since the run-time costs are orders of magnitude higher compared to matrix factorization for the same reconstruction error and compression ratio. Therefore, the application area for these techniques will be rather computer vision tasks and compression of even higher dimensional datasets where the factorization of additional modes amortizes but *not* interactive rendering of material appearance.

In order to further improve the performance of matrix factorization based compression techniques we introduced in Chapter 5 an algorithm which efficiently computes local coordinate systems for each sampled surface point of the BTF data and thereby improves the correlation between individual data vectors. This improved correlation results in a reduced rank of the co-variance matrix which means that less data has to be stored. Our method can be also interpreted as a general multi-view photo-metric stereo technique which does not depend on commonly used assumptions like diffuse reflectance.

Part II: Editing

The second part of this thesis dealt with another important aspect which might be not as obvious as data compression but is of almost equal practical relevance: editing of appearance data. Since measurement is an expensive process and in a practical workflow requirements can change frequently, 3D modeling artists need ways of editing the structure and reflectance of the measured materials. For example in one application the measured leather material should appear a little more glossy, while in another scenario the circular color patterns of a measured textile material should be square and so on. For this purpose we developed the BTF-Analogy technique which was presented in Chapter 8. It allows to transfer edits made on a single proxy image, e.g. a depth map of the material, to the whole dataset by learning correspondences between the low-dimensional proxy (the feature-of-interest map) and the high-dimensional BTF. The method was made interactive by exploiting state-of-art texture synthesis algorithms.

The BTF-Analogies also form the base of meaningful BTF interpolation presented in Chapter 9 which allows to change the reflectance behavior of BTFs in a completely data-driven way which means without any simplifying assumptions about the reflectance, e.g. in form of a parametric BRDF model. Together with a procedural model for the material structure a complete model of a particular material class based on real measurements can be assembled.

11.2 Future Work

In practical applications the creation of material appearance is still dominated by reflectance models, procedural shaders and standard textures. Nevertheless, the demand for high-quality measured appearance is growing because these techniques promise solutions for the productivity problem of Computer Graphics as mentioned in the introduction of this thesis. We hope, that this thesis acts here as another jigsaw piece which helps improving the applicability of highly productive representations like the BTF in practice. Of course, we are still only at the beginning of a process and many open research questions in the field BTF compression and editing await answering.

Part I: Compression

Most of the currently known compression techniques for BTFs have been adapted from other research fields like machine learning or image compression. What is

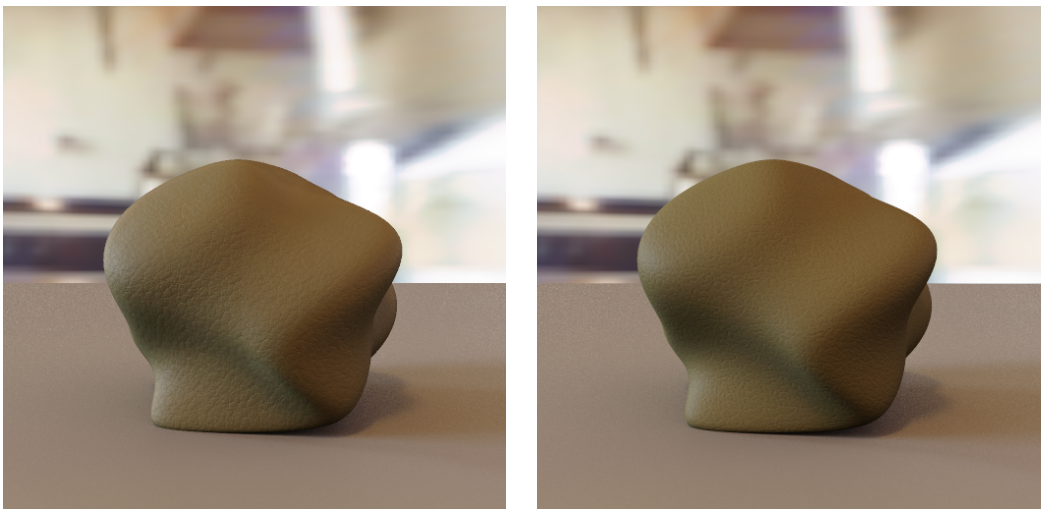


Figure 11.2: Comparing a real BTF with a completely synthetic model. What components make a material appear realistic? Is the BTF more realistic than the synthetic model?

still missing is a unique psycho-visual analysis for material appearance comparable, e.g. to the psycho-acoustic analysis known from audio compression. We have taken a first step towards this direction in [GMSK08] where psycho-visual techniques from video compression have been transferred to BTF compression.

In a similar direction points also another question which already has been brought up in our work about hybrid models for car-paint [RMS+08] – how many *real* or image-based data is needed and what parts of the appearance can be represented by a parametric model? This aspect touches the question about the *building blocks* of material appearance. Figure 11.2 shows two images of a blob-like object where one image was rendered with a measured material while the other image used a completely synthetic representation. Since the synthetic material model allows controlling its parts it is possible to add and leave certain parts of the material representation. Then it can be determined how these parts affect the perception in comparison to the real reference material.

It is not such a big surprise that many of the open questions in visual data compression are related to visual perception, since the size of our models and scenes is reaching a critical size were "brute force" techniques alone do not suffice anymore and an in-depth understanding of what parts of a scene are actually perceived by humans and which can be neglected becomes worthwhile.

The same can thing can be said about the light transport processes going on in the material. Starting from our work about data-driven local coordinate systems it would be interesting to investigate how real geometry can be traded for image-

based measurements and when do traditional representations like displacement maps and BRDFs suffice. How important is the simulation of higher order light transport and in what cases do we get away with approximations? An important issue here will be also the transition between scales since more than fifteen years after Becker and Max's seminal paper [BM93] there still does not seem to exist a satisfying answer to the question: what is the optimal multi-scale representation? An interesting starting point for further research might be the integration of measured light transport like BTFs into hierarchical pre-computed representations of light transport like the Meshless Hierarchical Light Transport recently introduced by Lehtinen et al. [LZT+08].

Part II: Editing

There is a multitude of possible directions for future research in material editing.

An obvious extension of our approach would be the integration of models for other interesting material classes like stone, wood or cloth where powerful procedural models are already available. In order to deal with materials consisting of several different substrates, e.g. with different colors, it will be also fruitful to combine our technique with the BTF selection operators introduced in [KBD07].

It will also be interesting to compare our results with simulated data or even to incorporate simulated materials to increase the expressiveness of the material database without the need for additional expensive measurements.

On the long run a generative model, probably based in parts on a database of measurements, which will allow to efficiently create photo-realistic and multi-scale instances of materials using an intuitive set of parameters will be the ultimate goal.

BIBLIOGRAPHY

[AB91] E. H. Adelson and J. R. Bergen. The plenoptic function and the elements of early vision. *M. Landy and J. A. Movshon, (eds) Computational Models of Visual Processing*, 1991.

[AP08] Xiaobo An and Fabio Pellacini. Appprop: all-pairs appearance-space edit propagation. *ACM Trans. Graph.*, 27(3):1–9, 2008.

[AS00] Michael Ashikhmin and Peter Shirley. An anisotropic Phong BRDF model. *Journal of Graphics Tools: JGT*, 5(2):25–32, 2000.

[Ash01] Michael Ashikhmin. Synthesizing natural textures. In *Symposium on Interactive 3D Graphics*, pages 217–226, 2001.

[BK07] Brett W. Bader and Tamara G. Kolda. Matlab tensor toolbox, version 2.2. `http://csmr.ca.sandia.gov/~tgkolda/TensorToolbox/`, January 2007.

[Bli78] James F. Blinn. Simulation of wrinkled surfaces. In *Proceedings of SIGGRAPH 78*, pages 286–292. ACM Press, 1978.

[BM93] Barry G. Becker and Nelson L. Max. Smooth transitions between bump rendering algorithms. In *SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 183–190, New York, NY, USA, 1993. ACM.

[BN76] James F. Blinn and Martin E. Newell. Texture and reflection in computer generated images. *Commun. ACM*, 19(10):542–547, 1976.

[Bra03] M. Brand. Fast online svd revisions for lightweight recommender systems. In *SIAM International Conference on Data Mining (SDM)*, May 2003.

[CBCG02] W-C. Chen, J-Y. Bouguet, M. H. Chu, and R. Grzeszczuk. Light field mapping: Efficient representation and hardware rendering of surface light fields. *Proceedings of ACM SIGGRAPH 2002*, 2002.

[Com74] L. Comtet. *Advanced Combinatorics*. Reidel, Dordrecht, 1974.

[Coo84] Robert L. Cook. Shade trees. *SIGGRAPH Comput. Graph.*, 18(3):223–231, 1984.

[CT81] Robert L. Cook and Kenneth E. Torrance. A reflectance model for computer graphics. In *SIGGRAPH '81: Proceedings of the 8th annual conference on Computer graphics and interactive techniques*, pages 307–316, New York, NY, USA, 1981. ACM Press.

[CWH93] Michael F. Cohen, John Wallace, and Pat Hanrahan. *Radiosity and realistic image synthesis*. Academic Press Professional, Inc., San Diego, CA, USA, 1993.

[DG98] P. Debevec and S. Gortler. Image-based modeling and rendering, 1998.

[DHT+00] P. Debevec, T. Hawkins, C. Tchou, H.-P. Duiker, W. Sarokin, and M. Sagar. Acquiring the reflectance field of a human face. *Proceedings of ACM SIGGRAPH 2000*, 2000.

[Dis98] Jean-Michel Dischler. Efficiently rendering macrogeometric surface structures using bi-directional texture functions. In *Rendering Techniques 98*. Springer Computer Science, 1998.

[DMM03] I. S. Dhillon, S. Mallela, and D. S. Modha. Information-theoretic co-clustering. In *Proceedings of The Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining(KDD-2003)*, pages 89–98, 2003.

[DvGNK97] Kristin J. Dana, Bram van Ginneken, Shree K. Nayar, and Jan J. Koenderink. Reflectance and texture of real-world surfaces. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 151–157, 1997.

[EL99] Alexei A. Efros and Thomas K. Leung. Texture synthesis by non-parametric sampling. In *Proceedings of the International Conference on Computer Vision-Volume 2*, page 1033. IEEE Computer Society, 1999.

[EMP+94] David Ebert, Kent Musgrave, Darwyn Peachey, Ken Perlin, and Worley. *Texturing and Modeling: A Procedural Approach*. Academic Press, October 1994. ISBN 0-12-228760-6.

[FC89] R. T. Frankot and R. Chellappa. A method for enforcing integrability in shape from shading algorithms. In B. K. P. Horn and M. J. Brooks, editors, *Shape from Shading*, pages 89–122. MIT Press, Cambridge, MA, 1989.

[FCGH08] J. Filip, M.J. Chantler, P.R. Green, and M. Haindl. A psychophysically validated metric for bidirectional texture data reduction. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia 2008)*, 27(5), December 2008.

[FH04] J. Filip and M. Haindl. Non-linear reflectance model for Bidirectional Texture Function synthesis. In J. Kittler, M. Petrou, and M. Nixon, editors, *Proceedings of the 17th IAPR International Conference on Pattern Recognition*, pages 80–83, Los Alamitos, August 2004. IEEE.

[GCHS05] Dan B. Goldman, Brian Curless, Aaron Hertzmann, and Steven M. Seitz. Shape and spatially-varying brdfs from photometric stereo. In *ICCV*, pages 341–348, 2005.

[GG91] Allen Gersho and Robert M. Gray. *Vector quantization and signal compression*. Kluwer Academic Publishers, Norwell, MA, USA, 1991.

[GGSC96] Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F. Cohen. The lumigraph. *Computer Graphics*, 30(Annual Conference Series):43–54, 1996.

[GLD00] G. Getz, E. Levine, and E. Domany. Coupled two-way clustering analysis of gene microarray data. In *Proc. Natl. Acad. Sci. USA*, pages 12079–12084, 2000.

[GMSK08] Michael Guthe, Gero Mueller, Martin Schneider, and Reinhard Klein. BTF-CIELab: a perceptual difference measure for quality assessment and compression of BTFs. *Submitted to Computer Graphics Forum*, 2008.

[GTLL06] Gaurav Garg, Eino-Ville Talvala, Marc Levoy, and Hendrik P. A. Lensch. Symmetric photography: Exploiting data-sparseness in reflectance fields. In Tomas Akenine-Möller and Wolfgang Heidrich, editors, *Eurographics Workshop/ Symposium on Rendering*, pages 251–262, Nicosia, Cyprus, 2006. Eurographics Association.

[GTR+06] J. Gu, C. Tu, R. Ramamoorthi, P. Belhumeur, W. Matusik, and S. K. Nayar. Time-varying Surface Appearance: Acquisition, Modeling, and Rendering. *ACM Trans. on Graphics (also Proc. of ACM SIGGRAPH)*, Jul 2006.

[Hac99] Wolfgang Hackbusch. A sparse matrix arithmetic based on $\mathcal{H}$-matrices. Part I: Introduction to $\mathcal{H}$-matrices. *Computing*, 62:89–108, 1999.

[HB95] David J. Heeger and James R. Bergen. Pyramid-based texture analysis/synthesis. In *Proceedings of SIGGRAPH*, pages 229–238. ACM Press, 1995.

[HF03] M. Haindl and J. Filip. Fast BTF texture modelling. In M. Chantler, editor, *Texture 2003. Proceedings*, pages 47–52, Edinburgh, October 2003. IEEE Press.

[HF07] M. Haindl and J. Filip. Extreme compression and modeling of bidirectional texture function. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(10):1859–1865, October 2007.

[HFA04] M. Haindl, J. Filip, and M. Arnold. BTF image space utmost compression and modelling method. In J. Kittler, M. Petrou, and M. Nixon, editors, *Proceedings of the 17th IAPR International Conference on Pattern Recognition*, pages 194–197, Los Alamitos, August 2004. IEEE.

[HJ86] Roger A. Horn and Charles R. Johnson. *Matrix analysis*. Cambridge University Press, New York, NY, USA, 1986.

[HJO+01] Aaron Hertzmann, Charles E. Jacobs, Nuria Oliver, Brian Curless, and David H. Salesin. Image analogies. In Eugene Fiume, editor, *SIGGRAPH 2001, Computer Graphics Proceedings*, pages 327–340. ACM Press / ACM SIGGRAPH, 2001.

[HKL+99] Kenneth E. Hoff III, John Keyser, Ming Lin, Dinesh Manocha, and Tim Culver. Fast computation of generalized Voronoi diagrams using graphics hardware. *Computer Graphics*, 33(Annual Conference Series):277–286, 1999.

[HWL05] Pun-Mo Ho, Tien-Tsin Wong, and Chi-Sing Leung. Compressing the illumination-adjustable images with principal component analysis. *IEEE Transactions on Circuits and Systems for Video Technolog*, 15(3):355–364, 2005.

[IO06] Hayley N. Iben and James F. O'Brien. Generating surface crack patterns. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 177–185, Sept 2006.

[JMLH01] Henrik Wann Jensen, Stephen R. Marschner, Marc Levoy, and Pat Hanrahan. A practical model for subsurface light transport. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 511–518, New York, NY, USA, 2001. ACM.

[Jol86] I.T. Jolliffe. *Principal Component Analysis*. Springer-Verlag, 1986.

[Kaj85] James T. Kajiya. Anisotropic reflection models. In *SIGGRAPH '85: Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, pages 15–21, New York, NY, USA, 1985. ACM.

[Kaj86] James T. Kajiya. The rendering equation. In *Proceedings of SIGGRAPH*, pages 143–150. ACM Press, 1986.

[KBD07] Jan Kautz, Solomon Boulos, and Frédo Durand. Interactive editing and modeling of bidirectional texture functions. *ACM Trans. Graph.*, 26(3):53, 2007.

[KCODL06] Johannes Kopf, Daniel Cohen-Or, Oliver Deussen, and Dani Lischinski. Recursive wang tiles for real-time blue noise. *ACM Transactions on Graphics*, 25(3 (Proc. SIGGRAPH 2006)):509–518, 2006.

[KEBK05] Vivek Kwatra, Irfan Essa, Aaron Bobick, and Nipun Kwatra. Texture optimization for example-based synthesis. *ACM Transactions on Graphics, SIGGRAPH 2005*, pages 795–802, August 2005.

[KL97] Nandakishore Kambhatla and Todd K. Leen. Dimension reduction by local principal component analysis. *Neural Comput.*, 9(7):1493–1516, 1997.

[KM99] Jan Kautz and Michael McCool. Interactive Rendering with Arbitrary BRDFs using Separable Approximations. In *Tenth Eurographics Workshop on Rendering*, pages 281–292, 1999.

[KMBK03] Mellisa L. Koudelka, Sebastian Magda, Peter N. Belhumeur, and David J. Kriegman. Acquisition, compression and synthesis of bidirectional texture functions. In *Texture 2003*, 2003.

[KR03] P. Kostelec and D. Rockmore. Ffts on the rotation group. Technical Report 03-11-060, Santa Fe Institute's Working Paper Series, 2003.

[Kuh55] Harold W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistic Quarterly*, (2):83–97, 1955.

[LBAD+06] Jason Lawrence, Aner Ben-Artzi, Christopher DeCoro, Wojciech Matusik, Hanspeter Pfister, Ravi Ramamoorthi, and Szymon Rusinkiewicz. Inverse shade trees for non-parametric material representation and editing. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 25(3), July 2006.

[Leh07] Jaakko Lehtinen. A framework for precomputed and captured light transport. *ACM Trans. Graph.*, 26(4):13, 2007.

[LFTG97] Eric P. F. Lafortune, Sing-Choong Foo, Kenneth E. Torrance, and Donald P. Greenberg. Non-linear approximation of reflectance functions. In *Proceedings of SIGGRAPH*, pages 117–126. ACM Press/Addison-Wesley Publishing Co., 1997.

[LGC+05] H. Lensch, M. Gösele, Y.-Y. Chuang, T. Hawkins, S. Marschner, W. Matusik, and G. Müller. Realistic materials in computer graphics. In *Siggraph Courses*, 2005.

[LGM07] Hendrik Lensch, Michael Gösele, and Gero Müller. Capturing reflectance - from theory to practice. In *Eurographics Tutorials*, 2007.

[LH96] Marc Levoy and Pat Hanrahan. Light field rendering. *Computer Graphics*, 30(Annual Conference Series):31–42, 1996.

[LH06] Sylvain Lefebvre and Hugues Hoppe. Appearance-space texture synthesis. *ACM Trans. Graph.*, 25(3):541–548, 2006.

[LHZ+04] Xinguo Liu, Yaohua Hu, Jingdan Zhang, Xin Tong, Baining Guo, and Heung-Yeung Shum. Synthesis and Rendering of Bidirectional Texture Functions on Arbitrary Surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 10(3):278–289, 2004.

[LKG+03] Hendrik P.A. Lensch, J. Kautz, Michael Goesele, Wolfgang Heidrich, and Hans-Peter Seidel. Image-based reconstruction of spatial appearance and geometric detail. *ACM Transactions on Graphics*, 2(22):234–257, 2003.

[LMV00] Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. On the best rank-1 and rank-(r1,r2,. . .,rn) approximation of higher-order tensors. *SIAM J. Matrix Anal. Appl.*, 21(4):1324–1342, 2000.

[LS00] Daniel D. Lee and H. Sebastian Seung. Algorithms for non-negative matrix factorization. In *NIPS*, pages 556–562, 2000.

[LYS01] Xinguo Liu, Yizhou Yu, and Heung-Yeung Shum. Synthesizing bidirectional texture functions for real-world surfaces. In *Proceedings of SIGGRAPH*, pages 97–106. ACM Press, 2001.

[LZT+08] Jaakko Lehtinen, Matthias Zwicker, Emmanuel Turquin, Janne Kontkanen, Frédo Durand, François Sillion, and Timo Aila. A meshless hierarchical representation for light transport. *ACM Trans. Graph.*, 27(3), 2008.

[Mac67] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In L. M. Le Cam and J. Neyman, editors, *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. University of California Press, 1967.

[MBK05] G. Müller, G. H. Bendels, and R. Klein. Rapid synchronous acquisition of geometry and BTF for cultural heritage artefacts. In *The 6th International Symposium on Virtual Reality, Archaeology and Cultural Heritage (VAST)*, pages 13–20. Eurographics Association, Eurographics Association, November 2005.

[McA02] David McAllister. *A Generalized Representation of Surface Appearance*. PhD thesis, University of North Carolina, 2002.

[MCT+05] Wan-Chun Ma, Sung-Hsiang Chao, Yu-Ting Tseng, Yung-Yu Chuang, Chun-Fa Chang, Bing-Yu Chen, and Ming Ouhyoung. Level-of-detail representation of bidirectional texture functions for real-time rendering. In *I3D '05: Proceedings of the 2005 symposium on Interactive 3D graphics and games*, pages 187–194, New York, NY, USA, 2005. ACM.

[MKC+06] Tom Mertens, Jan Kautz, Jiawen Chen, Philippe Bekaert, and Frédo Durand. Texture transfer using geometry correlation. In *Proceedings of Eurographics Symposium on Rendering*, 2006.

[MLH02] D. McAllister, A. Lastra, and W. Heidrich. Efficient rendering of spatial bi-directional reflectance distribution functions. *Graphics Hardware 2002*, 2002.

[MMK03] Gero Müller, Jan Meseth, and Reinhard Klein. Compression and real-time Rendering of Measured BTFs using local PCA. In *Vision, Modeling and Visualisation 2003*, pages 271–280, November 2003.

[MMK04a] Jan Meseth, Gero Müller, and Reinhard Klein. Reflectance field based real-time, high-quality rendering of bidirectional texture functions. *Computers and Graphics*, 28(1):103–112, February 2004.

[MMK04b] Gero Müller, Jan Meseth, and Reinhard Klein. Fast Environmental Lighting for Local-PCA Encoded BTFs. In *Computer Graphics International 2004 (CGI2004)*, June 2004.

[MMS+05] G. Müller, J. Meseth, M. Sattler, R. Sarlette, and R. Klein. Acquisition, synthesis and rendering of bidirectional texture functions. *Computer Graphics Forum*, 24(1):83–109, March 2005.

[MPBM03] Wojciech Matusik, Hanspeter Pfister, Matt Brand, and Leonard McMillan. A data-driven reflectance model. *ACM Trans. Graph.*, 22(3):759–769, 2003.

[MPDW03] Vincent Masselus, Pieter Peers, Philip Dutré, and Yves D. Willems. Relighting with 4d incident light fields. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers*, pages 613–620, New York, NY, USA, 2003. ACM.

[MPDW04] Vincent Masselus, Pieter Peers, Philip Dutré, and Yves D. Willems. Smooth reconstruction and compact representation of reflectance functions for image-based relighting. In *Rendering Techniques*, pages 287–298, 2004.

[MPN+02] Wojciech Matusik, Hanspeter Pfister, Addy Ngan, Paul Beardsley, Remo Ziegler, and Leonard McMillan. Image-based 3d photography using opacity hulls. In *Proceedings of SIGGRAPH*, pages 427–437. ACM Press, 2002.

[MRP98] Gavin S. P. Miller, Steven M. Rubin, and Dulce Ponceleon. Lazy decompression of surface light fields for precomputed global illumination. In *Proceedings of EGRW '98*, pages 281–292, 1998.

[MSK06] G. Müller, R. Sarlette, and R. Klein. Data-driven local coordinate systems for image-based rendering. *Computer Graphics Forum*, 25(3), September 2006.

[MSK07] G. Müller, R. Sarlette, and R. Klein. Procedural editing of bidirectional texture functions. In J. Kautz and S. Pattanaik, editors, *Eurographics Symposium on Rendering 2007*. The Eurographics Association, June 2007.

[MSRB07] Dhruv Mahajan, Ira Kemelmacher Shlizerman, Ravi Ramamoorthi, and Peter Belhumeur. A theory of locally low dimensional light transport. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers*, page 62, New York, NY, USA, 2007. ACM.

[MZD05] Wojciech Matusik, Matthias Zwicker, and Frédo Durand. Texture design using a simplicial complex of morphable textures. *ACM Trans. Graph.*, 24(3):787–794, 2005.

[NBB04] Shree K. Nayar, Peter N. Belhumeur, and Terry E. Boult. Lighting sensitive display. *ACM Trans. Graph.*, 23(4):963–979, 2004.

[NDM05] Addy Ngan, Frédo Durand, and Wojciech Matusik. Experimental analysis of brdf models. In *Proceedings of the Eurographics Symposium on Rendering*, pages 117–226. Eurographics Association, 2005.

[Nic65] F. E. Nicodemus. Directional reflectance and emissivity of an opaque surface. *Applied Optics*, 4(7):767–, 1965.

[NNJ05] Ko Nishino, Shree K. Nayar, and Tony Jebara. Clustered blockwise pca for representing visual data. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(10):1675–1679, 2005.

[NO77] F. E. Nicodemus and Others. *Geometrical Considerations and Nomenclature for Reflectance*. US Dept. of Commerce, National Bureau of Standards: for sale by the Supt. of Docs., US Govt. Print. Off., 1977.

[PB96] J.E. Proctor and P.Y. Barnes. Nist high accuracy reference reflectometer-spectrophotometer. *J. Res. Natl. Inst. Stand. Technol.*, 101(5):619–627, 1996.

[PH04] Matt Pharr and Greg Humphreys. *Physically Based Rendering: From Theory to Implementation*. Morgan Kaufmann, 2004.

[Pho75] Bui Tuong Phong. Illumination for computer generated pictures. *Communications of the ACM*, 18(6):311–317, 1975.

[PL07] Fabio Pellacini and Jason Lawrence. Appwand: editing measured materials using appearance-driven optimization. *ACM Trans. Graph.*, 26(3):54, 2007.

[POJ05] Fábio Policarpo, Manuel M. Oliveira, and ao L. D. Comba Jo˙ Real-time relief mapping on arbitrary polygonal surfaces. In *I3D '05: Proceedings of the 2005 symposium on Interactive 3D graphics and games*, pages 155–162, New York, NY, USA, 2005. ACM.

[RM00] Roerdink and Meijster. The watershed transform: Definitions, algorithms and parallelization strategies. *FUNDINF: Fundamenta Informatica*, 41, 2000.

[RMS+08] M. Rump, G. Müller, R. Sarlette, D. Koch, and R. Klein. Photo-realistic rendering of metallic car paint from image-based measurements. *Computer Graphics Forum*, 27(2), 2008.

[Row98] Sam Roweis. Em algorithms for pca and spca. In *in Advances in Neural Information Processing Systems*, pages 626–632. MIT Press, 1998.

[RTG97] Holly E. Rushmeier, Gabriel Taubin, and André Guéziec. Applying shape from lighting variation to bump map capture. In *Proceedings of EGRW '97*, pages 35–44. Springer-Verlag, 1997.

[Rus98] Szymon Rusinkiewicz. A new change of variables for efficient BRDF representation. In G. Drettakis and N. Max, editors, *Rendering Techniques '98*, pages 11–22, New York, NY, 1998. Springer Wien.

[Sch94] Christophe Schlick. An inexpensive brdf model for physically-based rendering. *Comput. Graph. Forum*, 13(3):233–246, 1994.

[SG92] Thomas W. Sederberg and Eugene Greenwood. A physically based approach to 2-D shape blending. In *Proceedings of SIGGRAPH*, volume 26, pages 25–34, 1992.

[Sha48] Claude Elwood Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27, 1948.

[SKU08] L. Szirmay-Kalos and T. Umenhoffer. Displacement mapping on the GPU - State of the Art. *Computer Graphics Forum*, 27(1), 2008.

[SOF] http://www.cs.dartmouth.edu/∼geelong/soft/.

[SSK03] M. Sattler, R. Sarlette, and R. Klein. Efficient and realistic visualization of cloth. *Proceedings of the Eurographics Symposium on Rendering 2003*, 2003.

[Sta95] Jos Stam. Multiple scattering as a diffusion process. In *In Proc. of EGSR*, pages 41–50, 1995.

[SvBLD03] Frank Suykens, Karl vom Berge, Ares Lagae, and Philip Dutré. Interactive Rendering of Bidirectional Texture Functions. In *Eurographics 2003*, pages 463–472, September 2003.

[TCS06] Michael Goesele Tongbo Chen and Hans-Peter Seidel. Mesostructure from specularity. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1825–1832, 2006.

[TZL+02] Xin Tong, Jingdan Zhang, Ligang Liu, Xi Wang, Baining Guo, and Heung-Yeung Shum. Synthesis of bidirectional texture functions on arbitrary surfaces. In *Proceedings of SIGGRAPH*, pages 665–672. ACM Press, 2002.

[VT04] M. A. O. Vasilescu and Demetri Terzopoulos. Tensortextures: Multilinear image-based rendering. In *Proceedings of SIGGRAPH*, August 2004.

[WAA+00] Daniel N. Wood, Daniel I. Azuma, Ken Aldinger, Brian Curless, Tom Duchamp, David H. Salesin, and Werner Stuetzle. Surface light fields for 3D photography. In *Proceedings of SIGGRAPH*, pages 287–296, 2000.

[War92] Gregory J. Ward. Measuring and modeling anisotropic reflection. In *Proceedings of SIGGRAPH*, pages 265–272. ACM Press, 1992.

[WAT92] Stephen H. Westin, James Arvo, and Kenneth E. Torrance. Predicting reflectance functions from complex surfaces. In James J. Thomas, editor, *SIGGRAPH*, pages 255–264. ACM, 1992.

[WHON97] Tien-Tsin Wong, Pheng-Ann Heng, Siu-Hang Or, and Wai-Yin Ng. Image-based rendering with controllable illumination. In *Proceedings of EGRW '97*, pages 13–22. Springer-Verlag, 1997.

[WHZ+08] Li-Yi Wei, Jianwei Han, Kun Zhou, Hujun Bao, Baining Guo, and Heung-Yeung Shum. Inverse texture synthesis. *Proceedings of ACM SIGGRAPH 2008*, 2008.

[WL00] Li-Yi Wei and Marc Levoy. Fast texture synthesis using tree-structured vector quantization. In *Proceedings of SIGGRAPH*, pages 479–488. ACM Press/Addison-Wesley Publishing Co., 2000.

[WL03] Tien-Tsin Wong and Chi-Sing Leung. Compression of illumination-adjustable images. *IEEE Transactions on Circuits and Systems for Video Technology (special issue on Image-based Modeling, Rendering and Animation)*, 13(11):1107–1118, 2003.

[WMP+06] Tim Weyrich, Wojciech Matusik, Hanspeter Pfister, Bernd Bickel, Craig Donner, Chien Tu, Janet McAndless, Jinho Lee, Addy Ngan, Henrik Wann Jensen, and Markus Gross. Analysis of human faces using a measurement-based skin reflectance model. *ACM Trans. Graph.*, 25(3):1013–1024, 2006.

[Wor96] Steven Worley. A cellular texture basis function. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 291–294, New York, NY, USA, 1996. ACM Press.

[WTL+04] Xi Wang, Xin Tong, Stephen Lin, Shimin Hu, Baining Guo, and Heung-Yeung Shum. Generalized Displacement Maps . pages 227–233, 2004.

[WWS+05] Hongcheng Wang, Qing Wu, Lin Shi, Yizhou Yu, and Narendra Ahuja. Out-of-core tensor approximation of multi-dimensional matrices of visual data. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*, pages 527–535, New York, NY, USA, 2005. ACM.

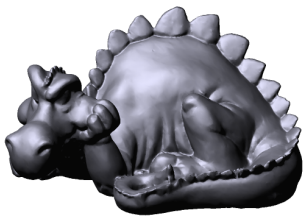
[WXC+08] Qing Wu, Tian Xia, Chun Chen, Hsueh-Yi Sean Lin, Hongcheng Wang, and Yizhou Yu. Hierarchical tensor approximation of multi-dimensional visual data. *IEEE Transactions on Visualization and Computer Graphics*, 14(1):186–199, 2008.

[ZDW+05] Kun Zhou, Peng Du, Lifeng Wang, Yasuyuki Matsushita, Jiaoying Shi, Baining Guo, and Heung-Yeung Shum. Decorating surfaces with bidirectional texture functions. *IEEE Transactions on Visualization and Computer Graphics*, 11(5):519–528, 2005.

[ZZV+03] Jingdan Zhang, Kun Zhou, Luiz Velho, Baining Guo, and Heung-Yeung Shum. Synthesis of progressively-variant textures on arbitrary surfaces. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers*, pages 295–302, New York, NY, USA, 2003. ACM.

DATA SOURCES

The following freely available data used in this thesis have been taken from external sources.



Dragon Model
UTIA, AS CR Prague
`http://ro.utia.cz/demos/drak/`

Vine St. Kitchen Light Probe
©1999 Paul Debevec
`http://www.debevec.org/Probes/`

Campus at Sunset Light Probe
©1999 Paul Debevec
`http://www.debevec.org/Probes/`

LIST OF PUBLICATIONS

M. RUMP, G. MÜLLER, R. SARLETTE, D. KOCH, AND R. KLEIN. Photorealistic rendering of metallic car paint from image-based measurements. In *Computer Graphics Forum*, 27(2), 2008.

G. MÜLLER, R. SARLETTE, AND R. KLEIN. Procedural editing of bidirectional texture functions. In *Eurographics Symposium on Rendering* 2007, June 2007.

G. MÜLLER, R. SARLETTE, AND R. KLEIN. Data-driven local coordinate systems for image-based rendering. In *Computer Graphics Forum*, 25(3), September 2006.

J. MESETH, G. MÜLLER, R. KLEIN, F. RÖDER AND M. ARNOLD. Verification of Rendering Quality from Measured BTFs. In *Proceedings of the 3rd Symposium on Applied Perception in Graphics and Visualization*, July 2006.

G. MÜLLER, G. H. BENDELS, AND R. KLEIN. Rapid synchronous acquisition of geometry and btf for cultural heritage artefacts. In *The 6th International Symposium on Virtual Reality, Archaeology and Cultural Heritage (VAST)*, 13–20. November 2005.

A. NICOLL, J. MESETH, G. MÜLLER, R. KLEIN. Fractional Fourier Texture Masks: Guiding Near-Regular Texture Synthesis. In *Computer Graphics Forum*, 24(3), September 2005.

G. MÜLLER, J. MESETH, M. SATTLER, R. SARLETTE, AND R. KLEIN. Acquisition, synthesis and rendering of bidirectional texture functions. In *Computer Graphics Forum*, 24(1):83–109, March 2005.

Z. NAGY, G. MÜLLER, R. KLEIN. Classification for Fourier Volume Rendering. In *Pacific Graphics 2004*, October 2004.

G. MÜLLER, J. MESETH, AND R. KLEIN. Fast Environmental Lighting for

Local-PCA Encoded BTFs. In *Computer Graphics International 2004*, June 2004.

J. MESETH, G. MÜLLER, AND R. KLEIN. Reflectance field based real-time, high-quality rendering of bidirectional texture functions. In *Computers and Graphics*, 28(1):103–112, February 2004.

G. MÜLLER, J. MESETH, AND R. KLEIN. Compression and real-time Rendering of Measured BTFs using local PCA. In *Vision, Modeling and Visualisation 2003*, pages 271–280, November 2003.